Policy Filtering with XACML

Quan Pham, Jason Reid, and Ed Dawson

Information Security Institute, Queensland University of Technology

126 Margaret Street, Brisbane QLD 4001, Australia

{*q.pham, jf.reid, e.dawson*}*@isi.qut.edu.au*

Abstract

This paper presents a modified approach to evaluate access control policy similarity and dissimilarity based on the proposal by Lin et al. (2007). Lin et al.'s policy similarity approach is intended as a filter stage which identifies similar XACML policies that can be analysed further using more computationally demanding techniques based on model checking or logical reasoning. This paper improves the approach of computing similarity of Lin et al. and also proposes a mechanism to calculate a dissimilarity score by identifying related policies that are likely to produce different access decisions. Departing from the original algorithm, the modifications take into account the policy obligation, rule or policy combining algorithm and the operators between attribute name and value. The algorithms are useful in activities involving parties from multiple security domains such as secured collaboration or secured task distribution. The algorithms allow various comparison options for evaluating policies while retaining control over the restriction level via a number of thresholds and weight factors.

Key words: Similarity, Dissimilarity, Relatedness, Relevance, Policy Evaluation, Policy Management, XACML, Access Control.

1. Introduction

The provision of seamless access to services located across multiple security domains is an emerging demand. This trend is consolidated by the development of service oriented architecture (SOA) and the federated technologies from various industry organisations. One of the key objectives of these technologies is to improve the productivity and efficiency of a service by connecting and provisioning it to a much wider range of clients. The vision is that clients could be allowed to collaborate and interact by accessing services or distributed resources across systems while maintaining an appropriate security posture and at the same time minimising any impediments. This requires the security authorities of the involved systems to understand and be able to verify security credentials of users from outside their domains. As discussed in the research statement, in order to achieve this capability, the security authorities must be able to answer the following question: "Given the user's information, related security policies of other involved systems and its own security policies, should the request be honoured?".

To address this challenge, it is vital to construct a mechanism to compare the involved constraints (written in the form of a policy) from other security domains with the local policies so that the local authority can say *"the external policy P_1 is similar to the local policy P_2"*. There is no trivial answer for this question. The solutions requires a combination of approaches in which comparing the applicable security policies is considered one of the most fundamental. The comparison process can be light-weight with low computational effort (and correspondingly low accuracy) or computationally expensive with more accurate methods such as Boolean checking or semantic analysis. The former to acts as a filter to identify relevant policies for more rigorous but computationally demanding analysis. As there are a considerable number of approaches designed to address the problem of evaluating policy compatibility or equivalence from various fields, the issue of a light-weight filter has attracted little attention from researchers in the field. This paper intends to address this issue via an algorithm to filtering similar/dissimilar policies.

The core of this paper is a policy filtering algorithm which can identify similar/dissimilar policies based on the requirements of the security authorities. Specifically, this paper proposes a modified algorithm to

evaluate similarity score of two policies and an algorithm to evaluate dissimilarity score of two policies based on the relatedness score of two policies. In addition, factors such as operators and obligations will be taken into account if applicable to get a better similarity or dissimilarity score. The algorithm is based on the seminal work of Lin et al. [12]. This paper proposes an algorithm that is an improved version of Lin et al.'s proposal to calculate both the similarity and dissimilarity of policies. The new algorithm also distinguishes the concept of relatedness from similarity. This paper assumes that there is a mechanism already in place to map common attributes with different names into common name spaces [17]. The mapping of different attribute names pointing to the same attributes is out of scope in this paper. Some original notations of Lin et al. are modified to serve the changes of the original algorithm and to improve the readability. As a widely adopted, well developed and mature standard, XACML is chosen to represent policy information.

The core of this paper is a set policy filtering algorithms which can identify similar/dissimilar policies based on the requirements of the security authorities. Specifically, this paper proposes a modification to Lin et al.'s algorithm to evaluate similarity score of two policies and a new algorithm to evaluate dissimilarity score of two policies based on the relatedness score of two policies. In addition, factors such as operators and obligations will be taken into account if applicable to get a more accurate similarity or dissimilarity score. The algorithm is based on the seminal work of Lin et al. [12]. This paper proposes an algorithm that is an improved version of Lin et al.'s proposal since Lin et al. did not consider dissimilarity. The new algorithm also distinguishes the concept of relatedness from similarity. Like Lin et al., this paper assumes that there is a mechanism already in place to map common attributes with different names into a common name space [17]. Some original notations of Lin et al. are modified to serve the changes of the original algorithm and to improve readability. As a widely adopted, well developed and mature standard, XACML is chosen to represent policy information.

The contributions of this paper are:

- analysing the approach of Lin et al. and identifying existing issues;
- proposing an improved algorithm to address the issues by taking into account additional factors such as combining algorithms or operators;
- introducing the notations of relatedness and dissimilarity and proposing algorithms to calculate them.

The rest of the paper is organised as follows. Section 2 reviews the existing approaches as well as the related concepts. This section also identifies the research problem and discusses the approach for a solution. Section 3 to Section 10 present the formulas and algorithm to evaluate the relatedness, similarity and dissimilarity score. Section 11 presents a case study to illustrate how the algorithms is applied and provides a summary and evaluation of the algorithm and provides some critical discussion about the algorithm's approach. 12 concludes the paper.

2. Background and Research Problem

2.1. XACML Policy, Relatedness, Similarity and Dissimilarity

XACML is designed to address the demand for a policy language with rich expressiveness to carry policy information for different security architectures and models. XACML provides an expressive XML-based platform-independent language including a rich syntax and semantics. XACML can provide the means to construct a wide range of policies such as "arbitrary attributes in policies, role-based access control, security labels, time/date-based policies, indexable policies, deny policies or dynamic policies" [16]. XACML provides methods to evaluate and combine different rules and policies into a single set to evaluate against a request. In XACML, there are five basic components: the Context Handler, the Policy Decision Point (PDP), the Policy Enforcement Point (PEP), the Policy Information Point (PIP) and the Policy Administration Point (PAP). The Context Handler plays the role of a coordinator. For each request, the Context Handler collects the necessary information from various sources such as the PIP, resources, or environments and send them to the PDP for decision making. The decision is then forwarded to the PEP for enforcement.

Figure 1 presents the structure of an XACML policy which consists of four major elements, namely Target, Rule Set, Rule-Combining Algorithm and Obligation Set(optional) [16]. The Target element of a Policy plays an important role in policy evaluation because it acts as a filter. The enclosed Rule Set will not be evaluated if the Target element of Policy is not matched with the request. Generally speaking, the value

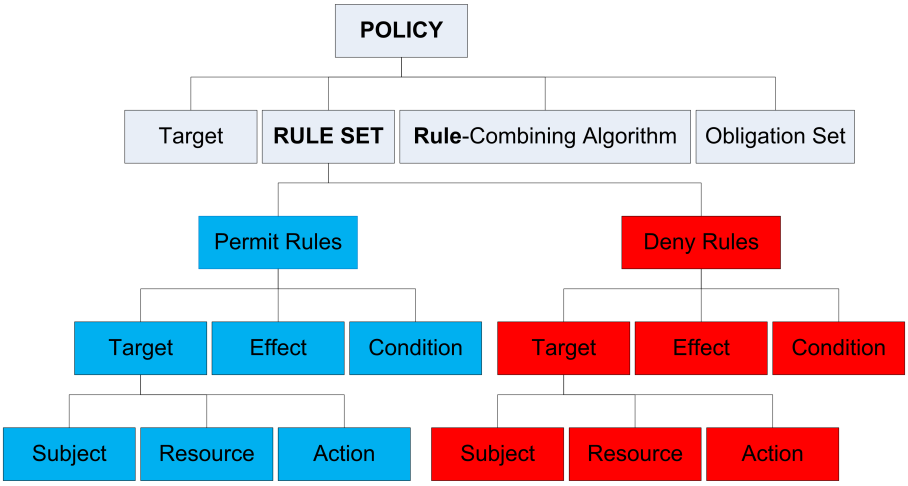

Figure 1: XACML Policy Structure.

of the Target element in the Policy is quite broad compared to the more specific Target element of a Rule. The Obligation Set of a Policy is a special element because it specifies an operation to be performed by the PEP in conjunction with the enforcement of an authorisation decision.

In XACML, the Rule-Combining Algorithm is very important. In XACML, each policy is a Boolean combination of a set of predicates. Due to this nature, it is essential for the policy processing engine to know how to combine the predicates together to get the overall evaluation result. The Rule-Combining Algorithm serves this purpose. The standard algorithms include Deny-overrides, Permit-overrides, First-applicable and Only-one-applicable. XACML can combine rules from multiple policies under various modes for example, deny-overrides or permit-overrides.

Each policy in XACML has a set of Permit or Deny rules. A Rule Set consists of one or more Rules. Each Rule in turn consists of a Target, an Effect element and a set of Conditions. The Target of a Rule element has the same structure as the Policy Target though they may be different in scope. It is usual that the Policy Target is broader. The Target element contains Subject, Resource and Action elements. However, in contrast with the Target value in a Policy or Policy Set, the value of the Rule's Target element tends to be more specific and is designed to capture the situation when the rule should be applied. A Condition element defines restrictions which the request must satisfy before a Permit or Deny decision can be made. An Effect element defines the effect (Permit/Deny) of a rule.

Attributes describe different characteristics of a Subject, Resource, Action or environment. They are named values with predefined types such as string or integer. Within a XACML policy, there are two types of attributes: content attributes and effect attributes. The content attributes define the content of the policies for example, Target, Subject, Resource, and Action. The effect attributes, on the other hand, define and orient the result (effect) of the policy, namely Obligation Set, Rule-Combining Algorithm, Condition and Effect. In general, each element of the policy or rule is defined as a set of predicates in the form of a tuple of:

$$\{\{attr\_name_1, attr\_op_1, attr\_val_1\}, ..., \{attr\_name_2, attr\_op_2, attr\_val_2\}\} \ . \tag{1}$$

For example, in the expression $AccessTime \geq$ *22:00*, the attribute name (*attr_name*) is *AccessTime*, operator (*attr_op*) is $\geq$ and attribute value (*attr_val*) is 22 : 00. Due to the difference in the nature of attribute value, predicates are divided into categorical predicates (predicates with attribute values which belong to certain domain-specific ontology) and numerical predicates (predicates with attribute values which belong to integer, real or date/time data types) [12].

As an interpretation of similarity, dissimilarity and relatedness, Budanitsky and Hirst (2005) asserted that "two concepts are close if their similarity or relatedness is high. Otherwise they are distant" [3]. In this paper, *two policies are defined to be related if they address the same Target.* Similarity is a special case of relatedness [3]. Two policies are close or not depending on the similarity between their policy elements. Two policies are objectively similar if they would yield, for a given request, the same effect on the same or

similar targets. Thus, similar policies have the following relationship tuple {*Same Target, Same Attribute, Same Effect*}. In this relationship tuple, Target is separated from other attributes because Target is the sole attribute to decide the relatedness.

The definition of dissimilarity is different. From this point of view, it is necessary to consider the factors that potentially create the difference in evaluating the dissimilarity. Under the context of a security policy, especially XACML, the factors are *Effect* and *Attribute*. From this observation, in this paper, there are two concepts of dissimilarity which are necessary to be distinguished and when computing the overall dissimilarity, it is also necessary to consider both of them.

- *Effect-based Dissimilarity:* In this case, the different results are created by the difference of Policy's Effects. This type of dissimilarity is based on the following relationship tuple: {*Same Target, Same Attribute,* ***Different Effect***}. This notation of dissimilarity captures pairs of policies that have similar attributes but are likely to generate contradictory results. For example, one policy says that the *"Access to Financial Database is Permitted from 8:00am"* and one policy indicated contradictorily *"Access to Financial Database is Denied from 8:00am"*. It should be note that Lin et al.'s approach would classify these two policies as potentially similar.

- *Attribute-based Dissimilarity:* In this case, the different results are created by the difference of Attributes of the two compared Policies. The relationship tuple, in this case, is {*Same Target,* ***Different Attribute****, Same Effect*}. The example above becomes *"Access to Financial Database is Permitted by Financial Staff only under the obligation of logging every access"* vs. *"Access to Financial Database is Permitted by Financial Staff and Audit Staff"*. It is important to note that the two policies can be considered as complementary to each other but essentially they are related but dissimilar. This type of dissimilarity is used to find pairs of policies that have similar Effects but still potentially generate different results. It should be note that the two policies must be related to be consider for dissimilarity evaluation.

As mentioned above, similarity is not the exactly same as relatedness, only a special case. Two related policies or rules are not necessarily similar. The relatedness, in fact, just concerns the Attributes of the policy, not the Effects. Therefore, from this perspective similarity and dissimilarity are two special cases of relatedness. Roughly, it can be considered that two dissimilar policies are related but the effects are, to some extent, contradictory. In addition, it is important to note that it seems that dissimilarity is the complement of the similarity (because both similarity and attribute-based dissimilarity look for policies with same Effects) and it may be argued that by computing *(1 - Similarity Score)* can achieve the dissimilarity score. However, essentially, it is not. Similarity and attribute-based dissimilarity are not complementary. Therefore, similarity and dissimilarity are also not complementary.

Figure 2 presents the relationship between similar, dissimilar and related policies in which the Grey area is the intersection of similar and dissimilar policies. The Grey area reflects policies whose similarity or dissimilarity can not be determined reliably by a light-weight comparison. More rigorous analysis is required. The intersection happens when there are a number of constraints that are different or are considered differently when computing similarity and dissimilarity.

It should be understood that the policy similarity should be evaluated based on not only the values of the attributes but also the operator between the attribute name and the value. In addition, the configurations that orient the result of interpreting such policies should be also considered. Therefore, when evaluating the similarity and dissimilarity, any algorithms must take into account the configuration factors such as Rule-Combing Algorithm or Obligation Set. The original algorithm of Lin et al. did not take into account these factors.

2.2. Policy Filtering and Related Scenarios

The following three scenarios are presented to illustrate the applications of a policy filtering mechanism and particular aspects of the policy filtering process.

Scenario 1 - Federation Forming (Static evaluation): Assume that there are a number of domains that are currently active in a federation. If a new domain wants to join the federation, it has to go through a number of steps to negotiate and achieve certain agreements either between the new domain and the existing member domains or between the new domain and the federation authorities. One aspect of the negotiation process is

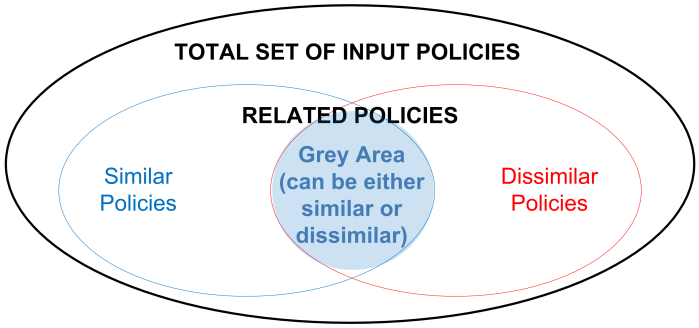

Figure 2: Similar, Dissimilar, and Related Policies.

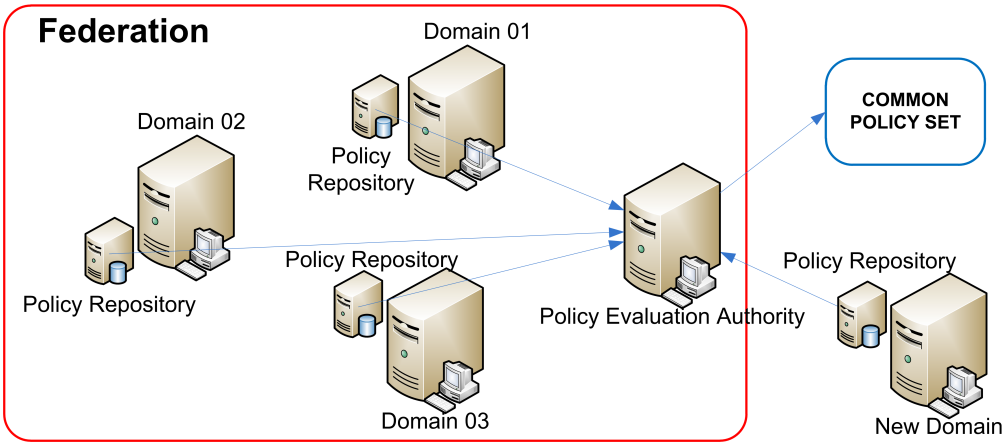

Figure 3: Scenario 1 - Policy Evaluation for Federation Establishment.

to achieve an understanding about a common security policy and so that no member of the federation would allow activities that would violate the common security context of the federation, especially activities that relate to cross security domain collaboration. To achieve this, the federation authority, the existing member domains and the new domain must evaluate the potentially involved security policies from all of the policy repositories. The purpose is to identify similar policies and possibly dissimilar policies for further negotiation in the common security context. The evaluation authority must retrieve and evaluate the involved policy sets from all involved domains (Figure 3). This is a static evaluation because it normally happens a limited number of times, for example, when initially joining the federation or when a member domain would like to introduce new policies.

Scenario 2 - Delegation Transaction (Dynamic evaluation of similar policies): Delegation usually happens in a collaborative activity when a user of one domain to pass on part of its privileges to another user in order to complete the necessary tasks (Figure 4). From the delegator's point of view, it is necessary for the delegator and possibly, its domain authority (Domain 02) to evaluate the involved policies of the delegatee's domain (Domain 01). If the policies of Domain 01 are similar, or in other words, allow similar activities, then the delegation should be allowed to happen.

Scenario 3 - Dynamic Collaborative Activity (Dynamic evaluation of dissimilar policies): As dissimilar policies are defined as policies with same target, similar attributes but a different effect, in some cases, it is important to quickly identify the dissimilar policies rather than similar policies. Consider a collaborative activity which involves access to a highly classified document. As this is a highly classified document, access should be tightly restricted. The authority tends to block a request rather than mistakenly allowing unauthorised access. In this case, the set of policies of the domain that hosts the document or the federation authority should be used as the basis to evaluate other policies. By looking for dissimilar policies, the authority can quickly rule out the access from unprivileged entities. The identification of dissimilar policies serves as a minimising false negative approach for environments with high security requirements. In this

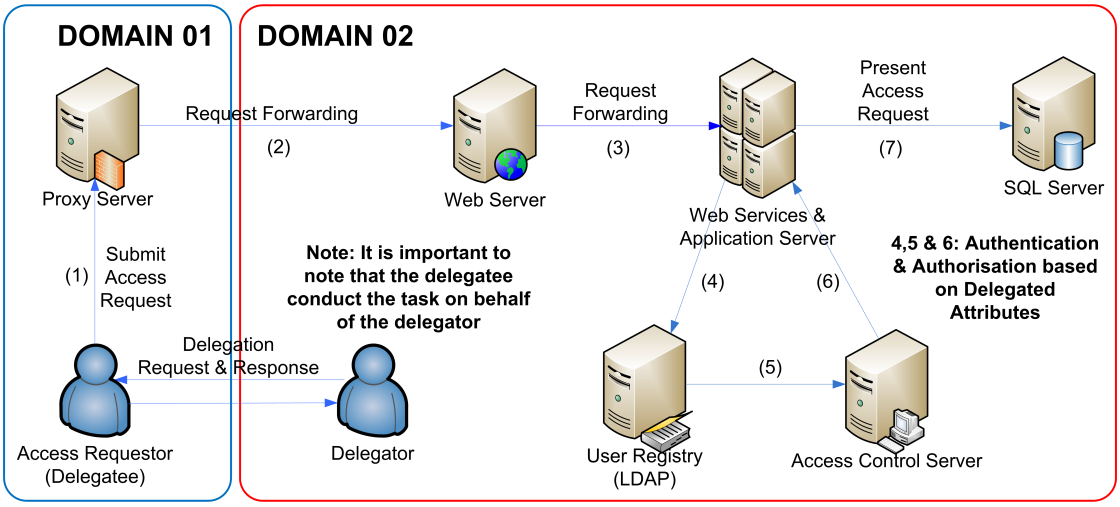

Figure 4: Scenario 2 - A Cross Domain Collaborative Activity via Delegation.

case, it may be still necessary to evaluate the similar policies later for other purposes.

Via these scenarios, it can be seen that there are a number of situations in which it is necessary to have a mechanism to efficiently identify the potential similar or dissimilar policies.

2.3. Related Works

In highly distributed environments as federated systems, it is not uncommon that a user from one domain may want to access services located on a different security domains. To authorise this type of request, the involved authorities may often need to evaluate not just their security policies but also the security policies from the domain of the user who initiates the access request. Also, they often need to compare these policies against each other to evaluate the security implications. The problem of comparing access control policies attracts a number of inputs from the research community. Most approaches are based on the principles of ontology and model checking [12, 13].

The most relevant work is the work of Lin et al [12] and the follow-up work, namely EXAM model [11]. EXAM allows checking similarity/dissimilarity using a combination of Multi-Terminal Binary Decision Diagram (MTBDD) and SAT-solver (Boolean Satisfiability) techniques. Backes et al. [2] propose an algorithm for checking the enhancement of privacy policies in an enterprise. The main objective of the algorithm is to check if one policy is a "subset" of another. Mazzoleni et al. [14] proposed a mechanism to address the issue of policy integration in the XACML model. The model is limited to working with a set of policies. The approach is also applicable for elements with same attributes as its main objective is to identify, for each attribute, which policy specifies the most restrictive conditions. Koch et al. [10] propose a uniform framework for comparing different policy models. Graph transformations are used to represent policy change and integration. Guelev et al. [9] also present a formal notation for expressing access control policies and queries which can evaluate access control policies written in different existing languages.

In terms of role-based systems, Fisler et al. [7] developed Margrave for analysing role-based access-control policies written in XACML. Margrave represents policies using MTBDD. The model is designed to verify policy properties and analyse the difference between versions of policies. Ahmed et al. [1] propose a model for role-based Computer Supported Cooperative Work (CSCW) to do similar tasks. The work allows expressing and verifying security constraints using finite-state model checking.

From the ontology perspective, string-based, semantic-based and graph-based approaches are widely considered as the principle tool for addressing the issue of similarity. Notable works in the field are OWL-Lite Aligner (OLA) [6], Quick Ontology Mapping (QOM) [5], Similarity Flooding (SF) [15], Schema-based matching (S-Match) [8], Combination of Matching Algorithms (COMA) [4]. The common problem with most approaches is that they are too computationally demanding with NP-complete [12]. This makes the existing approaches inappropriate for highly dynamic collaborative activities which require considerable processing efforts.

Among the approaches mentioned above, Lin et al.'s approach is one of the most interesting solutions because their work provides an inexpensive algorithm to compute the similarity of two policies written in XACML format [12]. Rather than relying on a brute force approach which would compute policy compatibility on the basis of the entire respective policy bases of the affected domains, their approach introduces a much less expensive algorithm based on information retrieval principles [12]. Lin et al. intend their similarity measure to be used as a filter to identify a subset of policies that are likely to produce similar access decisions. This subset can then be analysed further using more computationally expensive methods such as model checking or logical reasoning. Their proposed method aims to make such analysis more computationally feasible by reducing the number of policies that need to be examined. A key aspect to note about Lin et al.'s proposal is that it is concerned with policy similarity, not difference. They seek to identify policies that will produce the same or similar access decisions. In a subsequent work, Lin et al. classify this as a common property query which they distinguish from a discrimination query [11]. The latter identifies policies that affect similar subjects and resources but produce different access decisions. It is designed for a 2-stage approach: the first stage employs a light-weight filter to exclude obvious unrelated policies; the second stage is more comprehensive and involved a more computationally expensive policy analyser (EXAM [11]) for a further comparison.

The work of Lin et al. measure the similarity based on the approximation of the relationships between the sets of permit/deny rules of the two policies. The similarity score is a value between 0 and 1. This approach evaluates two policies by comparing permit rules of one policy with permit rules of the other. The same method is applied for deny rules. The similarity score obtained between the rules is then used to find one-to-many mappings (the Φ mappings) for each rule in the two policies. The Φ mapping is formed by mapping each permit rule on one policy to the permit rules on the other policy and vice versa. The similar method of mapping is applied for deny rules. By using the Φ mapping, the similarity score of a rule and a policy is computed. The purpose is to find out how similar a rule is with respect to the entire policy with a set of similar rules in the other policy. Then the scores of this type are aggregated and the average is the rule similarity score (with respect to the other entire policy). The same process is repeated for each rule in both permit and deny rule sets. Finally, by combining the similarity scores for permit and deny rule sets between two policies, the overall similarity score is obtained. The most computationally expensive operation of the approach is sorting and comparing attributes with same names. By comparing each attribute and moving to the next, the approach is similar to the heap sort algorithm. The complexity of the original algorithm is $O(n_a log(n_a) + n_a c)$, with n_a is the number of attributes in a rule and c is the approximated constant time to compare two attribute values [12]. If the number of attributes is not too large, then the average case complexity are $O(n_a log(n_a))$.

When evaluating the similarity score of permit/deny rule, only Target, Resource, Action and Condition elements are taken into account. The main reason is that this approach wants to minimise the potential false negatives. Similarity score for rules are obtained by basing the similarity on rule attributes which are divided into categorical and numerical types (i.e. failing to identity policies that are actually similar). Details of Lin et al.'s formulae and algorithm are presented in Appendix A.

2.4. Problems, Motivations and Approach

This section shows some limitations with the current approach of Lin et al. due to the lack of consideration of certain XACML elements and policy structure which potentially effect the final evaluation. This section also discusses the motivation and approach for further development of Lin et al.'s work.

Firstly, the work of Lin et al. provides a good approach to measure the similarity of the policies. As a light-weight evaluation method, Lin et al.'s approach is a good candidate for the first stage of the whole policy filtering process. The main issue is that when calculating the similarity, the original algorithm does not take into account the relatedness of the compared policies. This problem is reflected in how the similarity score of Target elements at Policy level are handled. In the case study of the original paper, the similarity score of the Target element is considered equally with the scores of other elements such as Permit Rule Set and Deny Rule Set. This configuration reflects that the importance of the Target element over other elements, as the sole factor to determine relatedness, is ignored. This is not a big problem. However, it shows that the original approach, to some extent, fail to recognise the concept of relatedness. The Policy's Target acts as a filter to determine whether an access request should be evaluated against the Rule's Targets. If a request does not match the Policy's Target, the Rule will be ignored. Therefore, in practice, rule similarity only

matters if Policies' Targets are related. If they are unrelated, there is no point to calculate the rule similarity (a waste of computational effort). If the two policies have a similarity score of Targets of 0.1, it should be assumed that they are written for two different things and so any similarity in the rules is a coincidence which should be disregarded. In the case study, the weight factors for Target, Permit Rule Set and Deny Rule Set are assigned equally (1/3 for each of them) [12]. It is argued that the evaluation of similarity should be conditioned on the relatedness score of the compared policies exceeding a certain threshold and the weight factor for Target element must be higher than other elements. The general philosophy behind Lin et al.'s work appears to be that potentially matched policies should not be ignored. Thus, there is a safety constraint that governs the way in which the original algorithm is designed. The impact of this constraint is that some factors such as Obligation Set or Rule-Combining Algorithm are ignored to elevate the similarity score to a safe level without more rigorous analysis. Basically, it is because it is not safe and reliable to determine the similarity of these factors. For example, effects of policies with the Combining Algorithm elements of *deny-override* vs. *first-applicable* can be similar depending on the order of rule elements, but rule orders is not considered in Lin et al.'s approach.

When comparing the similarity, the original algorithm of Lin et al. ignore certain rule elements, to avoid missing potential similar policies (and to minimise false negatives). However, when evaluating the dissimilarity, a different approach must be applied. In this paper, when calculating dissimilarity score, every possible factor must be taken into account to avoid missing potential dissimilar policies.

Therefore, the similarity/dissimilarity score should consider the operator used to specify/compare attribute values. The operator between the attribute name and value is also important as the operator actually defines the nature of the whole rule or policy. For example, it is obvious that two rules with the following conditions $AccessTime \geq 22{:}00$ vs. $AccessTime \leq 22{:}00$ are different. In fact, it is not sufficient and does not accurately reflect the nature of similarity of two elements, if only the attributes and values are evaluated as in the original algorithm of Li et al. Therefore, to overcome this shortcoming, in addition to the differences of value, the similarity of the operator between attribute name and value should also be considered. However, in order to maintain the safety nature of the original algorithm (e.g. minimise the number of false negatives), a score is only assigned when there is a solid ground to calculate. When there is no solid ground to calculate a score, 1 is assigned as a way to acknowledge the potential similarity (Section 4). The same approach is applied for calculating dissimilarity score. However, in case of dissimilarity, the safety is interpreted as trying to identify as many dissimilar policies as possible and accepting the possibility of mis-recognising similar policies as dissimilar.

Secondly, the policy structure which was used to design the algorithm for similarity needs to consider the Rule-Combining Algorithm and the Obligation Set. As mentioned above, in addition to a Target and a Rule Set, a Policy Set or a Policy also contains an Obligation Set. While this element does not contain the actual policy information related to the request, the involved resources or effects, it does generate a significant difference. Since the obligation defines certain operations that the PEP needs to complete as part of the authorisation decision, the Obligation Set does create a difference between the two compared policies. By performing the operations defined by the Obligation Set, the PEP can alter the security of the system. Consider two exactly matched policies, one from a more relaxed environment such as a commercial contractor, and one from a highly classified government agency. Due to strict security requirements, all activities of the agency must be monitored, logged and audited. The policy from the agency must include some obligations to this effect. This makes the two policies dissimilar in an important respect. Therefore, the similarity score of policies must include a score from Obligation Set.

Finally, as pointed out in the Scenario 3, it is important to consider a dissimilar score. The reason for an independent algorithm to evaluate dissimilarity is that the dissimilarity score can not be derived from the similarity score. It is important to point out that it is not true that the sum of similarity and dissimilarity equals to 1. In fact, it depends on the view of the algorithm (looking for similar or dissimilar policies), the percentage of the similar or dissimilar policies compared to the related policies are different. This leads to the issue that there will be many policies which, at the same time, can be considered as either similar or dissimilar (the *Grey Area* as in Figure 2). It is also intended to create a unified algorithm to calculate the similarity and dissimilarity. However, due to the potential confusion of policies in the Grey Area and the difference in dividing rules basing on the effect, it is problematic to employ this practice.

Therefore, to address those issues, this paper proposes:

- an algorithm to evaluate similarity score of two policies;

- an algorithm to evaluate dissimilarity score of two policies;
- that the Relatedness Score of two policies is taken into account before further calculating similarity or dissimilarity;
- factors such as operators and obligations may be taken into account if applicable to get a better similarity or dissimilarity score.

While the modified algorithm will take into account additional factors, the one-to-many mapping to compare a rule of one policy to the corresponding Rule Set of the other policy will be the same. In addition, the algorithms in this paper also operates based on the a common hierarchy in case of comparing two policies from two different domains (with two different hierarchies). The common hierarchy provides the basis for the algorithm to evaluate the similarity/dissimilarity of XACML elements. The following sections will discuss these issues in more detail and present formal definition and complete algorithms to achieve these objectives. Table 1 presents a modified set of notations for the new formulae and algorithms. The formulae in this paper should be interpreted based on the modified notations, not the Lin et al.'s original notations.

Notation	**Meaning**
P	Policy
RS	Rule Set
PR	Permit Rule Set
DR	Deny Rule Set
$\langle Effect \rangle$	Effect of rule or policy. The value can be P for Permit or D for Deny rule or policy.
$\langle Type \rangle$	Type of dissimilarity measure. The value can be A for Attribute-based or E for Effect-based dissimilarity evaluation.
$\langle Element \rangle$	Policy or Rule elements in which T is Policy Target; CA is Rule-Combining Algorithm; OS is Obligation Set; RS is Rule Set; t is Rule Target; c is Rule Condition; s is Rule Subject; r is Rule Resource and a is Rule Action.
r	Rule
a	Attribute
v	Attribute value
H	Height of a hierarchy
$\mathcal{M}_a$	Set of pairs of matching attribute names
$\mathcal{M}_v$	Set of pairs of matching attribute values
N_{PR}	Number of permit rules in a policy
N_{DR}	Number of deny rules in a policy
N_a	Number of attributes in an element
N_v	Number of values of an attribute
$SPath$	Length of shortest path of two categorical values
$w^{\langle Effect \rangle}_{\langle Element \rangle}$	Weight of similarity score of elements, where $\langle Element \rangle \in \{T, OS, RS, t, c, s, r, a\}$
R_{policy}	Relatedness score of policies
R_{rule}	Relatedness score of rules
S_{policy}	Similarity score of two policies
S_{rule}	Similarity score of two rules
$S^{P}_{rule-set}$	Similarity score of two permit rule sets
$S^{D}_{rule-set}$	Similarity score of two deny rule sets
$S^{\langle Effect \rangle}_{\langle Element \rangle}$	Similarity score of elements, where $\langle Element \rangle \in \{T, OS, RS, t, c, s, r, a\}$

Continued on next page

Table 1 – Continued from previous page

Notation	Meaning
$s_{cat-val}$	Similarity score of two categorical values
S_{cat}	Similarity score of two categorical predicates
$s_{num-val}$	Similarity score of two numerical values
S_{num}	Similarity score of two numerical predicates
s_{os-val}	Similarity score of obligation values
S_{os}	Similarity score of two obligation predicates
rs	Similarity score of a rule and a policy
ϵ	Rule similarity threshold
Φ	Similar rule mapping
$\delta_{cat-val}/\delta_{num-val}$ $/\delta_{os-val}$	Compensating score of unmatched categorical/numerical/obligation values when computing similarity.
D_{policy}	Overall dissimilarity score of two policies
$D_{policy}^{\langle Type\rangle}$	Attribute-based (A) or Effect-based (E) dissimilarity score of two policies.
D_{rule}	Dissimilarity score of two rules
$D_{rule-set}^{P}$	Dissimilarity score of two permit rule sets
$D_{rule-set}^{D}$	Dissimilarity score of two deny rule sets
$D_{\langle Element\rangle}^{\langle Type\rangle}$	Dissimilarity score of elements, where $\langle Element\rangle \in \{T, CA, OS, RS, t, c, s, r, a\}$
d_{cat}	Dissimilarity score of two categorical values
D_{cat}	Dissimilarity score of two categorical predicates
d_{num}	Dissimilarity score of two numerical values
D_{num}	Dissimilarity score of two numerical predicates
d_{os-val}	Dissimilarity score of obligation values
D_{os}	Dissimilarity score of two obligation predicates
$w_{\langle Element\rangle}^{\langle Type/Effect\rangle}$	Weight of dissimilarity score of elements, where $\langle Element\rangle \in \{T, CA, OS, RS, t, c, s, r, a\}$
$rd^{\langle Type\rangle}$	Dissimilarity score of a rule and a policy
Ω	Dissimilarity rule mapping
$\sigma^{\langle Type\rangle}$	Rule dissimilarity threshold and
$\theta_{cat-val}/\theta_{num-val}$ $/\theta_{os-val}$	Compensating score of unmatched categorical/numerical/obligation values when computing dissimilarity

Table 1: The Modified Notations.

3. Relatedness Score of Policies and Rules

As mentioned above, there is no point to investigate the similarity or dissimilarity of policies or rules, if these policies or rules are unrelated, or in other words, addressing different targets for policies and targets, subjects and resources for rules (excepts perhaps to identity potential policy errors). Therefore, it is necessary to calculate the relatedness score of policies or rules before further evaluating similarity or dissimilarity. In XACML, due to the difference in structure of Policy and Rule element, the formula to calculate the relatedness score is different for these two cases.

- Relatedness Score of Policies: Because the Policy element just has one Target element, the relatedness score of policies is equal to the similarity score of the Target element. Therefore, the relatedness score R_{Policy} of two policies are defined as below.

$$R_{policy}(P_1, P_2) = S_T(P_1, P_2)\ . \tag{2}$$

- Relatedness Score of Rules: The relatedness score of policies equals to the similarity score of the Target element which contains three sub-elements namely Subject, Resource and Action element. The relatedness score R_{rule} of two policies are defined as below.

$$R_{rule}(r_1, r_2) = S_t(r_1, r_2) \ . \tag{3}$$

The next sections will discuss in details how the similarity scores for Target element of Policy and Rule are calculated.

4. Similarity Score of Policy or Rule Elements

Lying at the core of the original approach of Lin et al. is the formulae to calculate the similarity between elements at both the policy and rule levels. Similar to the original algorithm, firstly, predicates for each rule are clustered based on attribute name. Then predicates with same attribute names are computed to get a similarity score. Finally, the score of each part of matching predicates are summarised to achieve the similarity score of the Rule element.

However, in the original approach, only the attribute name and the value of the element were taken into account. To overcome this issue, the similarity of the operator between attribute name and value of predicates is defined and evaluated. Due to the difference in the nature of attribute types, predicates are divided into categorical predicates (such as predicates with attribute types which belong to certain domain-specific ontology) and numerical predicates (predicates with attribute values which belong to integer, real or date/time data types) [12]. The similarity score of an element $S_{\langle Element \rangle}$ is defined as below.

$$S_{\langle Element \rangle}(r_i, r_j) = \begin{cases} \dfrac{\sum\limits_{(a_{1k}, a_{2l}) \in \mathcal{M}_a} S_{attr-type}(a_{1k}, a_{2l})}{max(N_{a_1}, N_{a_2})}, & N_{a_1} > 0 \text{ and } N_{a_2} > 0 \ ; \\ 1, & \text{otherwise} \ . \end{cases} \tag{4}$$

$$S_{attr-type}(a_{1k}, a_{2l}) = \begin{cases} S_{cat}(a_{1k}, a_{2l}), & \text{if } a_{1k} \text{ and } a_{2l} \text{ are categorical} \ ; \\ S_{num}(a_{1k}, a_{2l}), & \text{if } a_{1k} \text{ and } a_{2l} \text{ are numerical} \ . \end{cases} \tag{5}$$

The similarity score of two rules regarding the same element is denoted as $S_{\langle Element \rangle}$, where *Element* refers to *Target (t), Subject (s), Resource (r), Condition (c)* or *Action (a)*. The next section will discuss the formula for calculating similarity score S_{cat} for categorical predicates and S_{num} for numerical predicates in detail. $\mathcal{M}_a$ is a set of pairs of matching predicates with same attributes; a_{1k} and a_{2l} are attributes of rules r_{1i} and r_{2l}; and N_{a_1} and N_{a_2} are the numbers of distinct predicates in the two rules.

4.1. Similarity Score of Categorical Predicates

Unlike the original algorithm, the operator between attribute name and value is taken into account. The similarity score S_{cat} is defined as follows.

$$S_{cat}(a_1, a_2) = \frac{1}{3} \left[1 + \frac{\sum\limits_{(v_{1k}, v_{2l}) \in \mathcal{M}_v} s_{cat-val}(v_{1k}, v_{2l}) + \delta_{cat-val}}{max(N_{v_1}, N_{v_2})} + s_{cat-op} \right] \ . \tag{6}$$

In the above equation, 1 represents the value of the attribute names of the predicates because they are exactly matched. In other words, only predicates with exactly matched attribute names are evaluated.

$$\delta_{cat-val} = \begin{cases} \dfrac{\sum\limits_{(v_{1k}, -) \notin \mathcal{M}_v} \sum\limits_{l=1}^{N_{v_2}} s_{cat-val}(v_{1k}, v_{2l})}{N_{v_2}}, & N_{v_1} > N_{v_2} \ ; \\ \dfrac{\sum\limits_{(-, v_{2l}) \notin \mathcal{M}_v} \sum\limits_{k=1}^{N_{v_1}} s_{cat-val}(v_{1k}, v_{2l})}{N_{v_1}}, & N_{v_1} < N_{v_2} \ . \end{cases} \tag{7}$$

The similarity score $s_{cat-val}$ of hierarchical value is defined as below. The main difference between these formulae and those on the original algorithm is the operator between attribute name and value and the weight factors. To calculate the dissimilarity score of the attribute values, the approach of computing the compensating score ($\delta_{cat-val}$), tree traversal, $SPath$ and $Hcode$ in Lin et al.'s work is adapted [12].

$$s_{cat-val}(v_1, v_2) = 1 - \frac{SPath(v_1, v_2)}{2H} \ . \tag{8}$$

s_{cat-op} is the similarity score of the operators. The idea to calculate the similarity score of operators is that if a full match or a partial match is achieved (for example, *Designation belong_to*{*QPIFStaff, ESStaff*} vs. *Designation=QPIFStaff*), then the full score of 1 is assigned to reflect the safety nature as discussed in Section 2.4. Otherwise, *nil* is assigned.

While a full match is relatively simple to determine (two operators are identical), it is more difficult to determine an appropriate values for a partial match. In XACML, values of attributes are categorised into the following data types namely *string, boolean, integer, double, time, date, dateTime, dayTimeDuration, yearMonthDuration, anyURI, hexBinary, base64Binary, rfc882Name* and *x500Name*. Associating with each data type, there are a finite predefined set of operators as well as finite set of operators that can be used if these data types are presented in a form of bag and set. Because the common types of categorical predicates are basically *string, anyURI, rfc882Name* and *x500Name*, therefore, only operators for string data type should be applied, namely, *string-equal, string-greater-than, string-greater-than-or-equal, string-less-than* and *string-less-than-or-equal* or any applicable operators for bag, set and regular expression. A partial match is determined if the two operators contain equality (*string-equal, string-greater-than-or-equal* and *string-less-than-or-equal*) or type of same inequality (*string-greater-than* and *string-greater-than-or-equal*) or (*string-less-than* and *string-less-than-or-equal*). As XACML defines different match or regular expression operator for *anyURI, rfc882Name* and *x500Name* data types, there will be no partially match for these data types. Therefore, a partial match is achieved if operator 1 and 2 are not identical and are elements of the following sets {*string-equal, string-greater-than-or-equal, string-less-than-or-equal*}, {*string-greater-than, string-greater-than-or-equal*} and {*string-less-than, string-less-than-or-equal*}.

The similarity score of the operators s_{cat-op} is defined as below.

$$s_{cat-op}(a_1, a_2) = \begin{cases} 1, & \text{if the relationships are fully or partial matched ;} \\ 0, & \text{otherwise .} \end{cases} \tag{9}$$

4.2. Similarity Score of Numerical Predicates

As there is no hierarchical relationship among numerical predicates, it is logical to consider the difference of the values as the basis for similarity evaluation. However, as discussed above, the operator between the attribute name and value is also important as the operator actually defines the nature of the whole rule or policy. Therefore, in addition to the differences of value, the similarity of the operator must be also considered. Similarly to the formulae to calculate the similarity score of categorical predicates, the similarity score S_{num} for the whole attribute is defined in the following equation.

$$S_{num}(a_1, a_2) = \frac{1}{3}\left[1 + \frac{\sum_{(v_{1k}, v_{2l}) \in \mathcal{M}_v} s_{num-val}(v_{1k}, v_{2l}) + \delta_{num-val}}{max(N_{v_1}, N_{v_2})} + s_{num-op}\right] \ . \tag{10}$$

Similar to the formula for categorical values, in the above equation, 1 represents the value of the attribute names of the predicates because they are exactly matched. In other words, only predicates with exactly matched attribute names are evaluated.

The original formula of Lin et al. for $s_{num-val}$ does not exactly reflect the similarity of two values. It can be seen that the less difference between two values creates the less similarity score which is incorrect (should be high similarity scores). The definition of $\mathcal{M}_v$ is similar to the definition for categorical attributes.

$$s_{num-val}(v_1, v_2) = 1 - \frac{|v_1 - v_2|}{max(v_1, v_2)} \ . \tag{11}$$

The above equation is designed to reflect the similarity of two numerical values. Assuming that $v_1 > v_2$, then the equation becomes $1 - (v_1 - v_2)/v_1$ or v_2/v_1. If $v_1 \gg v_2$, then $v_2/v_1 \to 0$. Therefore, the similarity score is close to 0 which correctly reflects that $v_1 \gg v_2$.

Unlike categorical predicates, the value of a numerical predicate is basically *integer, double, time, date, dateTime, dayTimeDuration* or *yearMonthDuration*. Therefore, only operators associated to these data types should be considered. A partial match is achieved if operator 1 and 2 are not identical and are elements of the following sets {*type-equal, type-greater-than-or-equal, type-less-than-or-equal*}, {*type-greater-than, type-greater-than-or-equal*} and {*type-less-than, type-less-than-or-equal*} (*type* is of one the above data types above).

The similarity score of the operator s_{num-op} is defined as follows.

$$s_{num-op}(a_1, a_2) = \begin{cases} 1, & \text{if the relationships are fully matched ;} \\ 0, & \text{otherwise .} \end{cases} \tag{12}$$

Similar to the categorical predicate, $\delta_{num-val}$ is calculated in the same way. It is defined by the equation below.

$$\delta_{num-val} = \begin{cases} \dfrac{\sum_{(v_{1k},-)\notin\mathcal{M}_v} \sum_{l=1}^{N_{v_2}} s_{num-val}(v_{1k}, v_{2l})}{N_{v_2}}, & N_{v_1} > N_{v_2} \ ; \\ \dfrac{\sum_{(-,v_{2l})\notin\mathcal{M}_v} \sum_{k=1}^{N_{v_1}} s_{num-val}(v_{1k}, v_{2l})}{N_{v_1}}, & N_{v_1} < N_{v_2} \ . \end{cases} \tag{13}$$

5. Policy Similarity Measure

The policy similarity is a score to evaluate the level of match of two policies. The score is between 0 and 1. The enhanced algorithm, in this section, utilises the same one-to-many comparison technique to compute the Φ mapping for each rule in the two policies as in the original algorithm. However, unlike the original algorithm, the modification will also take a threshold factor (ϵ) as decision making factor.

The policy similarity S_{policy} is presented by the following equation.

$$\begin{gathered} S_{policy}(P_1, P_2) = w_T S_T(P_1, P_2) + w_{RS} S_{RS}(P_1, P_2) + w_{OS} S_{OS}(P_1, P_2) \ ; \\ \text{where } w_T + w_{OS} + w_{RS} = 1 \ . \end{gathered} \tag{14}$$

In the above equation, S_T, S_{RS} and S_{OS} are the similarity scores of Target's Obligation Set and Rule Set respectively and w_T, w_{RS} and w_{OS} are the weight factors of these elements. These weigh factors are adjustable to reflect the importance of the elements of a policy.

Among the three similarity score of Target, Obligation Set and Rule Set, the similarity scores of Targets and Rule Set are the most important because targets and rules are the foundation elements of a policy. The next sections will discuss in detail how to calculate the similarity scores for all elements. It is important to note that in case of similarity measure, the Rule-Combing Algorithm should not be considered because all Rule-Combining Algorithms supported by XACML are able to generate similar effect depending on how the subsequent rules/policies are organised. So, to preserve the safety property as discussed in Section 2.4 , it is safe to exclude the Rule-Combining Algorithm from similarity measure.

5.1. Similarity Score of Targets

In general, to compute the similarity score of Target elements, an appropriate equation must be applied depending on the nature of the type of the Target element's predicates. Generally speaking, the Target element at the Policy level in most policies is usually hierarchical and therefore, categorical (Section 4.1). Unlike the original algorithm, after calculating the similarity score of Targets, the enhanced algorithm uses the score to make the decision whether to continue or not. The threshold value (ϵ) must be applied for this purpose. If the relatedness score of two policies is below the threshold, the algorithm will be terminated and the similarity score of the two policies or rules will be zero. That implies the Equation 14 should be expressed in the below form.

$$S_{policy}(P_1, P_2) = \begin{cases} w_T S_T(P_1, P_2) + w_{RS} S_{RS}(P_1, P_2) + w_{OS} S_{OS}(P_1, P_2), & R_{policy} \geq \epsilon \ ; \\ 0, & \text{otherwise} \ . \end{cases} \tag{15}$$

This is to improve the overall efficiency of the algorithm as a lot of computational resource can be saved when evaluating a large number of policies.

5.2. Similarity Score of Obligation Sets

The Obligation Set is an optional element of a policy. However, this element could become quite important because the Obligation Set is usually utilised to perform post-authorisation activities such as logging for auditing purposes. An Obligation Set includes a number of obligations. Obligations accompany the Permit or Deny authorisation decision of the PDP and must be enforced by the PEP.

In order to measure the similarity, firstly the Obligation Set is divided into two subsets (based on the value of the *FulfillOn* element): one subset contains obligations for Permit decision and one subset for Deny decision. Then the similarity score S_{OS}^P and S_{OS}^D for Permit and Deny obligations must be calculated.

The similarity score S_{OS} of the Obligation Set is defined in the following equation below, where w_{OS}^P and w_{OS}^D are the weight factors for the two subset of obligations.

$$\begin{gathered} S_{OS}(P_1, P_2) = w_{OS}^P S_{OS}^P(P_1, P_2) + w_{OS}^D S_{OS}^D(P_1, P_2) \ ; \\ \text{where } w_{OS}^P + w_{OS}^D = 1 \ . \end{gathered} \tag{16}$$

Generally speaking, the Obligation Set value are usually text therefore the comparison should be just an approximation and string-based. For each Obligation Set, the *AttributeAssignment* element is considered. Then values with exactly matched *AttributeId* attributes are compared. Finally, the scores of each pair are summarised. Possibly, not all *AttributeId* are matched. Therefore, a penalty should be included. The penalty calculation is similar to the formula of categorical or numerical predicates. Then by applying the same technique as in Section 4.1 and 4.2, the similarity score of Obligation Set $S_{OS}^{\langle Effect \rangle}$ is defined as below.

$$S_{OS}^{\langle Effect \rangle}(P_1, P_2) = \begin{cases} \dfrac{\sum\limits_{(a_{1k}, a_{2l}) \in \mathcal{M}_a} S_{os}(a_{1k}, a_{2l})}{max(N_{a_1}, N_{a_2})}, & N_{a_1} > 0 \text{ and } N_{a_2} > 0 \ ; \\ 1, & \text{otherwise} \ . \end{cases} \tag{17}$$

Because the names of attributes are exactly matched in this case, so the similarity score of attribute name is 1. Also, it is important to note that only equality (=) operator is used to compare *AttributeIds* and their values, 1 will be assigned to signified the similarity of operators. Therefore, the similarity score of *AttributeIds* are defined as below. The formulae for attribute-based dissimilarity of categorical values is used to compute the scores.

$$S_{os}(a_1, a_2) = \frac{1}{3} \left[1 + \frac{\sum\limits_{(v_{1k}, v_{2l}) \in \mathcal{M}_a} s_{os-val}(v_{1k}, v_{2l}) + \delta_{os-val}}{max(N_{v_1}, N_{v_2})} + 1 \right] \ . \tag{18}$$

$$\delta_{os-val} = \begin{cases} \dfrac{\sum\limits_{(v_{1k}, -) \notin \mathcal{M}_v} \sum\limits_{l=1}^{N_{v_2}} s_{os-val}(v_{1k}, v_{2l})}{N_{v_2}}, & N_{v_1} > N_{v_2} \ ; \\ \dfrac{\sum\limits_{(-, v_{2l}) \notin \mathcal{M}_v} \sum\limits_{k=1}^{N_{v_1}} s_{os-val}(v_{1k}, v_{2l})}{N_{v_1}}, & N_{v_1} < N_{v_2} \ . \end{cases} \tag{19}$$

Because the value is text, therefore, the operators for string data type should be applied, namely, *string-equal, string-greater-than, string-greater-than-or-equal, string-less-than*, and *string-less-than-or-equal*. A score of 1 should be applied for any of these operators.

$$s_{os-val}(v_1, v_2) = \begin{cases} 1, & \text{if any string operator is applied} \ ; \\ 0, & \text{otherwise} \ . \end{cases} \tag{20}$$

5.3. Similarity Score of Rule Sets

In order to calculate the similarity score of Rules in the Rule Set, the whole set will be divided into two subsets namely Permit Rule Set and Deny Rule Set based on the Effect of the Rules (Permit/Deny). Each single rule in each policy is then compared with a rule in another policy that has the same effect, and a similarity score of two rules is obtained. The similarity score obtained between the rules is then used to find one-to-many mappings (Φ mappings) for each rule in the two policies. There are four mappings to consider namely $PR_1(PR_2)$ - Φ_1^P, $PR_2(PR_1)$ - Φ_2^P, $DR_1(DR_2)$ - Φ_1^D and $DR_2(DR_1)$ - Φ_2^D. For example, the mapping $PR_1(PR_2)$ maps each Permit Rule r_{1i} in P_1 with one or more Permit Rules r_{2j} in P_2. Similarly the mapping $PR_2(PR_1)$ maps each Permit Rule r_{2j} in P_2 with one or more Permit Rules r_{1i} in P_1. For each rule in P_1, the Φ mappings give similar rules in P_2 which satisfy certain similarity threshold and vice versa [12].

By adapting the approach of Lin et al., the similarity score of the Rule Set S_{RS} are calculated by dividing into two sub-scores namely S_{RS}^P (score for rule with permit effect) and S_{RS}^D (score of rules with deny effect).

$$\begin{gathered} S_{RS} = w_P S_{RS}^P + w_D S_{RS}^D \ ; \\ \text{where } w_P + w_D = 1 \ . \end{gathered} \tag{21}$$

Then, by using the Φ mapping and calculating the one-to-many mapping for each permit and deny rule, S_{RS}^P and S_{RS}^D are defined in the below equations where S_{Rule} is the similarity score of each pair of rules as the result of the Φ mapping.

$$S_{RS}^P = \frac{\sum_{i=1}^{N_{PR_1}} rs_1 i + \sum_{j=1}^{N_{PR_2}} rs_2 j}{N_{PR_1} + N_{PR_2}} \ . \tag{22}$$

$$S_{RS}^D = \frac{\sum_{i=1}^{N_{DR_1}} rs_1 i + \sum_{j=1}^{N_{DR_2}} rs_2 j}{N_{DR_1} + N_{DR_2}} \ . \tag{23}$$

$$rs_{1i} = \begin{cases} \dfrac{\sum\limits_{r_j \in \Phi_1^P(r_{1i})} S_{rule}(r_{1i}, r_j)}{|\Phi_1^P(r_{1i})|}, & r_{1i} \in PR_1 \ ; \\ \dfrac{\sum\limits_{r_j \in \Phi_1^D(r_{1i})} S_{rule}(r_{1i}, r_j)}{|\Phi_1^D(r_{1i})|}, & r_{1i} \in DR_1 \ . \end{cases} \tag{24}$$

$$rs_{2j} = \begin{cases} \dfrac{\sum\limits_{r_i \in \Phi_2^P(r_{2j})} S_{rule}(r_{2j}, r_i)}{|\Phi_2^P(r_{2j})|}, & r_{2j} \in PR_2 \ ; \\ \dfrac{\sum\limits_{r_i \in \Phi_2^D(r_{2j})} S_{rule}(r_{2j}, r_i)}{|\Phi_2^D(r_{2j})|}, & r_{2j} \in DR_2 \ . \end{cases} \tag{25}$$

The procedure for computing rule similarity is based on the idea of evaluating all similar policies in the Φ mapping. The procedure is illustrated in Algorithm A.3 (Lin et al's original procedure in Appendix A). The next section shows how to calculate S_{rule}.

5.4. Similarity Score of Rules

This is the step to calculate similarity scores for each individual pair of rules in the Φ mapping. It is fairly simple because the total score is a sum of similarity scores of Rules' Targets and Conditions. It is important to note that because the rule's Effect is already utilised to divide the whole Rule Set into Permit and Deny subsets, it should be not counted again in this step. As a rule has Targets and Conditions and a rule's Target consists of Subjects, Resources and Actions, the formulae to calculate similarity scores S_{rule} are defined as below where w_t, w_c, w_s, w_r and w_a are the weight factors of the corresponding elements.

$$S_{rule}(r_i, r_j) = \begin{cases} w_t S_t(r_i, r_j) + w_c S_c(r_i, r_j), & R_{rule} \geq \epsilon \ ; \\ 0, & \text{otherwise} \ . \end{cases} \quad \text{where } w_t + w_c = 1 \ . \tag{26}$$

$$S_t(r_i, r_j) = w_s S_s(r_i, r_j) + w_r S_r(r_i, r_j) + w_a S_a(r_i, r_j) \ ; \quad \text{where } w_s + w_r + w_a = 1 \ . \tag{27}$$

As Target, Condition, Subject, Resource and Action consist of either categorical or numerical predicates, the same formulae and techniques can be applied to calculate similarity scores (Section 4.1 and 4.2).

Unlike the original algorithm, it is important to note that the rule's Target element should have higher weight factor than other elements. Similar to the policy's Target element, a checking against the threshold value should be included to improve the efficiency of the overall algorithm.

5.5. Overall Algorithm

This enhanced algorithm is proposed based on the Lin et al.'s algorithm and the observations about its shortcomings. The modified algorithm for Policy Similarity Measure is presented in Algorithm 5.1. Lines 1 to 7 are the additional steps to take into account the Target, Rule-Combining Algorithm and Obligation Set element. Line 2 is specifically included to reduce the processing time by checking the similarity score of Target against the threshold value. The rest is quite similar to the original algorithm. However, the main difference is that the formulae to calculate similarity score of rules' elements are changed to reflect the nature of the operator between attribute name and value. Line 8 categorises rules in P_1 and P_2 based on their effects. Lines 9 to 14 compute the similarity score S_{rule} for each pair of rules in P_1 and P_2. Lines 15 to 18 compute the Φ mappings. Lines 19 to 33 use the Φ mappings to calculate the rule set similarity scores and sum them up to achieve the similarity scores S_{RS} for Rule Sets. Finally, line 34 calculates the overall similarity score by adding the scores of Target, Rule-Combing Algorithm, Obligation Set and Rule Set. The procedure to compute Φ mapping (ComputePhiMapping) and rule similarity (ComputeRuleSimilarity) are same as in Lin et al.'s original procedures. The original procedures are presented in Algorithm A.2 and A.3 (Appendix A).

The current algorithm has the potential to address the concern about dissimilarity as discussed in Section 2.4. As mentioned above, the original algorithm and the modified algorithm employ common property query for similarity analysis. However, in some cases, dissimilarity may be important as the system authority may want to know whether the difference of dissimilar policies has any negative impacts [11]. The attribute-based dissimilarity can use the same Φ mapping. The effect-based dissimilarity of rules can be achieved by inverting the mapping of rule. Instead of creating Permit-Permit ($PR(PR)$) and Deny-Deny ($DR(DR)$) mappings, the permit rules of P_1 are mapped with Deny rules of P_2 and vice versa to create $PR(DR)$ and $DR(PR)$ mappings. The next sections will discuss dissimilarity issue in more detail.

6. Attribute-based Dissimilarity Score of Policy or Rule Elements

Similar to the formulae to calculate similarity score, the attribute-based dissimilarity score is also calculated based on the same set of elements namely, Target, Rule-Combining Algorithm, Obligation Set and Rule Set. Sub-Elements of these elements, if applicable, are categorised as categorical or numerical predicates. In this case, *every possible element must be taken into account.* This is contradictory to the algorithm to calculate similarity score as the similarity algorithm just takes into account a minimal set of elements to minimise false negatives.

In this case, the dissimilarity of the operator between attribute name and value of predicates is defined and evaluated. The score of two rules regarding the same element is denoted as $D_{\langle Element \rangle}$, where *Element* refers to *Target (t), Subject (s) , Resource (r), Condition (c)* or *Action (a).* $D_{\langle Element \rangle}$ is computed by comparing the corresponding predicate sets in two rules. Firstly, the predicates for each rule element according to the attribute names are clustered. Secondly, the predicates in the two rules whose attribute names match exactly are identified and the score is computed based on their attribute values. Finally, scores of each pair of matching predicates are summarised to obtain the attribute-based dissimilarity score of the rule element. The attribute-based dissimilarity score for an element $D_{\langle Element \rangle}$ is defined as below.

Algorithm 5.1: *EnhancedPolicySimilarityMeasure*(P_1, P_2) Algorithm - The Enhanced Algorithm for Policy Similarity Measure.

input : P_1 with n rules $\{r_{11}, r_{12}, ..., r_{1n}\}$; P_2 with m rules $\{r_{21}, r_{22}, ..., r_{2m}\}$.

output: Policy similarity $S_{policy}(P_1, P_2)$.

`/* Compute Relatedness Score */`

$R_{policy}(P_1, P_2)$

`/* Comparing the Relatedness Score with threshold */`

if $R_{policy} < \epsilon$ **then**

  $S_{policy}(P_1, P_2) = 0$

  return $S_{policy}(P_1, P_2)$

end

`/* Compute Similarity Scores of Obligation Sets */`

Categorise Obligation Sets based on FulfillOn value (Permit/Deny)

$S_{OS}(P_1, P_2)$

Categorise rules in P_1 and P_2 based on their effects. Let $PR_1(PR_2)$ and $DR_1(DR_2)$ denote the set of permit and deny rules respectively in $P_1(P_2)$.

`/* Similarity scores for each rule in` P_1 `and` P_2 `using new formulae */`

foreach *rule* $r_{1i} \in PR_1$ **do**

  foreach *rule* $r_{2j} \in PR_2$ **do** $S_{rule}(r_{1i}, r_{2j})$

end

foreach *rule* $r_{1i} \in DR_1$ **do**

  foreach *rule* $r_{2j} \in DR_2$ **do** $S_{rule}(r_{1i}, r_{2j})$

end

`/* Compute` Φ `mappings */`

$\Phi_1^P \leftarrow ComputePhiMapping(PR_1, PR_2, \epsilon)$

$\Phi_2^P \leftarrow ComputePhiMapping(PR_2, PR_1, \epsilon)$

$\Phi_1^D \leftarrow ComputePhiMapping(DR_1, DR_2, \epsilon)$

$\Phi_2^D \leftarrow ComputePhiMapping(DR_2, DR_1, \epsilon)$

`/* Compute the Rule Set similarity scores */`

foreach *rule* $r_{1i} \in P_1$ **do**

  if $r_{1i} \in PR_1$ **then**

    $rs_{1i} \leftarrow ComputeRuleSimilarity(r_{1i}, \Phi_1^P)$

  else if $r_{1i} \in DR_1$ **then**

    $rs_{1i} \leftarrow ComputeRuleSimilarity(r_{1i}, \Phi_1^D)$

end

foreach *rule* $r_{2j} \in P_2$ **do**

  if $r_{2j} \in PR_2$ **then**

    $rs_{2j} \leftarrow ComputeRuleSimilarity(r_{2j}, \Phi_2^P)$

  else if $r_{1i} \in DR_1$ **then**

    $rs_{2j} \leftarrow ComputeRuleSimilarity(r_{2j}, \Phi_2^D)$

end

$S_{RS}^P \leftarrow$ average of rs of permit rules

$S_{RS}^D \leftarrow$ average of rs of deny rules

$S_{RS} = w_P S_{RS}^P + w_D S_{RS}^D$

`/* Compute the overall similarity score */`

$S_{policy}(P_1, P_2) = w_T S_T(P_1, P_2) + w_{OS} S_{OS} + w_{RS} S_{RS}(P_1, P_2)$

return $S_{policy}(P_1, P_2)$

$$D_{\langle Element\rangle}(r_i, r_j) = \begin{cases} \dfrac{\sum\limits_{(a_{1k},a_{2l})\in\mathcal{M}_a} D^A_{attr-type}(a_{1k}, a_{2l})}{max(N_{a_1}, N_{a_2})}, & N_{a_1} > 0 \text{ and } N_{a_2} > 0\ ; \\ 1, & \text{otherwise}\ . \end{cases} \tag{28}$$

$$D_{attr-type}(a_{1k}, a_{2l}) = \begin{cases} D^A_{cat}(a_{1k}, a_{2l}), & \text{if } a_{1k} \text{ and } a_{2l} \text{ are categorical}\ ; \\ D^A_{num}(a_{1k}, a_{2l}), & \text{if } a_{1k} \text{ and } a_{2l} \text{ are numerical}\ . \end{cases} \tag{29}$$

In this equation, $\mathcal{M}_a$ is a set of pairs of matching predicates with same attribute name, a_{1k} and a_{2l} are the attributes of the involved rules and $N_(a_1)$ and $N_(a_2)$ are the numbers of distinct predicates in the two rules. The next section will discuss the formula for calculating dissimilarity score D_{cat} for categorical predicates and D_{num} for numerical predicates in detail.

6.1. Attribute-based Dissimilarity Score of Categorical Predicates

Unlike the original algorithm, the operator between attribute name and value is taken into account. To calculate the dissimilarity score of the attribute values, the approach of tree traversal, $SPath$ and $Hcode$ in Lin et al.'s work is adapted. However, in terms of dissimilarity, the further two values are on the common hierarchy tree, the more dissimilar they should be, therefore, the dissimilarity score of $d_{cat-val}$ of hierarchical values is defined as below.

$$d_{cat-val}(v_1, v_2) = \frac{SPath(v_1, v_2)}{2H}\ . \tag{30}$$

The dissimilarity score D_{cat} is defined in the below equations. As only predicates with exactly matched attribute names are evaluated, the dissimilarity score of the attribute names of the predicate should be 0.

$$\begin{aligned} D_{cat}(a_1, a_2) &= \tfrac{1}{3}\left[0 + \frac{\sum\limits_{(v_{1k},v_{2l})\in\mathcal{M}_v} d_{cat-val}(v_{1k}, v_{2l}) + \theta_{cat-val}}{max(N_{v_1}, N_{v_2})} + d_{cat-op}\right] \\ &= \tfrac{1}{3}\left[\frac{\sum\limits_{(v_{1k},v_{2l})\in\mathcal{M}_v} d_{cat-val}(v_{1k}, v_{2l}) + \theta_{cat-val}}{max(N_{v_1}, N_{v_2})} + d_{cat-op}\right]\ . \end{aligned} \tag{31}$$

$$\theta_{cat-val} = \begin{cases} \dfrac{\sum\limits_{(v_{1k},-)\notin\mathcal{M}_v} \sum\limits_{l=1}^{N_{v_2}} d_{cat-val}(v_{1k}, v_{2l})}{N_{v_2}}, & N_{v_1} > N_{v_2}\ ; \\ \dfrac{\sum\limits_{(-,v_{2l})\notin\mathcal{M}_v} \sum\limits_{k=1}^{N_{v_1}} d_{cat-val}(v_{1k}, v_{2l})}{N_{v_1}}, & N_{v_1} < N_{v_2}\ . \end{cases} \tag{32}$$

The idea to calculate the dissimilarity score d_{cat-op} for the operator is that if a full match is achieved then the full score of *nil* is assigned. If a partial match is achieved (for example, *Designation belong_to*{*QPIFStaff, ESStaff*} vs. *Designation=QPIFStaff*), then a full score is assigned to reflect the partial overlap of the operator which potentially represents a difference. Otherwise, 1 is also assigned. While a full match is relatively simple to determine (two operators are identical), it is more difficult to determine an appropriate values for a partial match.

By using a similar approach as in Section 4.1, the dissimilarity score of the operators d_{cat-op} is defined as below.

$$d_{cat-op}(a_1, a_2) = \begin{cases} 0, & \text{if the relationships are fully matched}\ ; \\ 1, & \text{otherwise}\ . \end{cases} \tag{33}$$

In the equation above, $\theta_{cat-val}$ is the compensation factor for unmatched attributes. The idea is that if there are some matched predicates, the score should be reduced to reflect the similarity (so less dissimilar).

Unlike the similarity algorithm, $\mathcal{N}_v$ is a set of pairs of unmatched pairs of values which have the following properties:

- If $v_{1k} \neq v_{2l}$, then $(v_{1k}, v_{2l}) \in \mathcal{N}_v$.
- For pairs $v_{1k} = v_{2l}$, pairs contributing to the maximum sum of dissimilarity scores belong to $\mathcal{N}_v$.
- Each attribute value v_{1k} or v_{2l} occurs at most once in $\mathcal{N}_v$.

6.2. Attribute-based Dissimilarity Score of Numerical Predicates

As there is no hierarchical relationship among numerical predicates, it is logical to consider the difference of the values as the basic for dissimilarity evaluation. However, as discussed above, the operator between the attribute name and value is also important as the operator actually defines the nature of the whole rule or policy. Therefore, in addition to the differences of value, the dissimilarity of the operator must be also considered.

The dissimilarity score of two numerical values is basically the difference of two values:

$$d_{num-val}(v_1, v_2) = \frac{|v_1 - v_2|}{max(v_1, v_2)} \ . \tag{34}$$

The above equation is designed to reflect the dissimilarity of two numerical values. Assuming that $v_1 > v_2$, then the equation becomes $(v_1 - v_2)/v_1$ or $1 - v_2/v_1$. If $v_1 \gg v_2$, then $v_2/v_1 \to 0$. Therefore, the dissimilarity score is close to 1.

Similar to the formula for categorical values, only predicates with exactly matched attribute names are evaluated. Therefore, the value of the attribute names of the predicates should be 0. The dissimilarity score D_{num} for the whole attribute is defined in the equations below.

$$\begin{aligned} D_{num}(a_1, a_2) &= \frac{1}{3}\left[0 + \frac{\sum_{(v_{1k}, v_{2l}) \in \mathcal{M}_v} d_{num-val}(v_{1k}, v_{2l}) + \theta_{num-val}}{max(N_{v_1}, N_{v_2})} + d_{num-op}\right] \\ &= \frac{1}{3}\left[\frac{\sum_{(v_{1k}, v_{2l}) \in \mathcal{M}_v} d_{num-val}(v_{1k}, v_{2l}) + \theta_{num-val}}{max(N_{v_1}, N_{v_2})} + d_{num-op}\right] \ . \end{aligned} \tag{35}$$

$$\theta_{num-val} = \begin{cases} \dfrac{\sum_{(v_{1k}, -) \notin \mathcal{M}_v} \sum_{l=1}^{N_{v_2}} s_{num-val}(v_{1k}, v_{2l})}{N_{v_2}}, & N_{v_1} > N_{v_2} \ ; \\ \dfrac{\sum_{(-, v_{2l}) \notin \mathcal{M}_v} \sum_{k=1}^{N_{v_1}} s_{num-val}(v_{1k}, v_{2l})}{N_{v_1}}, & N_{v_1} < N_{v_2} \ . \end{cases} \tag{36}$$

By using a similar approach as in Section 4.2, the dissimilarity score of the operators d_{num-op} are defined as below.

$$d_{cat-op}(a_1, a_2) = \begin{cases} 0, & \text{if the relationships are fully or partial matched ;} \\ 1, & \text{otherwise .} \end{cases} \tag{37}$$

7. Policy Attribute-based Dissimilarity Measure

Policy attribute-based dissimilarity is a score to evaluate the level of match of two policies. The score is between 0 and 1. The algorithm in this section utilises the same one-to-many comparison technique to compute the Ω mapping for each rule in the two policies as in the original algorithm. The dissimilarity score is only evaluated if the relatedness score is larger than a threshold value (σ^A). At this stage, a relatedness score of Targets is also computed. The relatedness score is utilised to make the decision whether to continue

or not. The threshold value must be applied for this purpose. If the relatedness score of two policies is below the threshold, the algorithm will be terminated and the dissimilarity score of the two policies or rules will be one. This is to improve the overall efficiency of the algorithm as a lot of computational resource can be saved when evaluating large number of policies. The policy dissimilarity D^A_{policy} is presented by the below equation.

$D^A_{policy}(P_1, P_2) =$

$$\begin{cases} w^A_T D^A_T(P_1,P_2) + w^A_{CA} D^A_{CA}(P_1,P_2) + w^A_{OS} D^A_{OS}(P_1,P_2) + w^A_{RS} D^A_{RS}(P_1,P_2), & R_{policy} \geq \sigma^A \ ; \\ 0, & \text{otherwise} \ . \end{cases} \tag{38}$$
$$\text{where } w^A_T + w^A_{CA} + w^A_{OS} + w^A_{RS} = 1 \ .$$

D^A_T, D^A_{CA}, D^A_{OS}, and D^A_{RS} are the dissimilarity scores of Target, Rule-Combining Algorithm, Obligation Set and Rule Set respectively and w^A_T, w^A_{CA}, w^A_{OS}, and w^A_{RS} are the weight factors of these elements. These weigh factors are adjustable to reflect the importance of the four elements of a policy.

Among the four attribute-based dissimilarity score of Target, Rule-Combining Algorithm, Obligation Set and Rule Set, the scores of Targets and Rule Set are the most important because targets and rules are the foundation elements of a policy. It is important to note that, unlike the similarity measure, the Rule-Combining Algorithm is taken into account because it is a factor that potentially causes difference. The next sections will discuss in detail how to calculate the dissimilarity scores for all four elements.

7.1. Attribute-based Dissimilarity Score of Targets

In general, to compute the dissimilarity score of Target elements, an appropriate equation must be applied depending on the nature of the type of the Target element's predicates (categorical or numerical). Generally speaking, the Target element at the Policy level in most policies is usually hierarchical.

7.2. Attribute-based Dissimilarity Score of Rule-Combining Algorithms

In XACML, to achieve certain level of conflict resolution, a limited set of Rule-Combining Algorithm is provided. The Rule-Combining Algorithm set (CA) includes {*deny-overrides, permit-overrides, first-applicable, only-one-applicable, ordered-deny-overrides, order-permit-overrides*}.

Due to the discrete nature of this Rule-Combining Algorithm set and exact match of name of the Rule-Combining Algorithm, the operator between attribute name and value is not required. Therefore, the dissimilarity score D^A_{CA} of this element in the policy is simply defined as below.

$$D^A_{CA} = \begin{cases} 0, & \text{if the algorithms are matched} \ ; \\ 1, & \text{otherwise} \ . \end{cases} \tag{39}$$

The evaluation is fairly simple because generally speaking, each policy or rule has only one Rule-Combining Algorithm element. A full score should be assigned for anything rather than an identical match. Because even though the compared Rule-Combining Algorithms are partially matched, they may potentially yield the different result. For example, deny-overrides and first-applicable algorithm can produce either same or different effect given that the two policies have identical rules and structure them in an appropriate order.

7.3. Attribute-based Dissimilarity Score of Obligation Sets

The Obligation Set is an optional element of a policy. However, this element could become quite important because the Obligation Set element is usually utilised to perform post-authorisation activities such as logging for auditing purposes. An Obligation Set includes a number of obligations. Obligations accompany the Permit or Deny authorisation decision of the PDP and must be enforced by the PEP.

In order to measure the dissimilarity, firstly the Obligation Set is divided into two subsets (based on the value of the *FulfillOn* element): one subset contains Obligations for Permit decision and one subset for Deny decision. Then the dissimilarity score D^P_{OS} and D^D_{OS} for Permit and Deny obligations must be calculated.

The dissimilarity score D^A_{OS} of the Obligation element is defined in the following equation below where w^P_{OS} and w^D_{OS} are the weight factors for the two subset of obligations.

$$D^A_{OS}(P_1,P_2) = w^P_{OS} D^P_{OS}(P_1,P_2) + w^D_{OS} D^D_{OS}(P_1,P_2) \ ; \quad \text{where } w^P_{OS} + w^D_{OS} = 1 \ . \tag{40}$$

Generally speaking, the Obligation Set values are usually string therefore the comparison should be just approximation and string-based. For each Obligation Set, the *AttributeAssignment* element is considered. Then values with exactly matched *AttributeId* attributes are compared. Finally, the scores of each pair are summarised. Possibly, not all *AttributeIds* are matched. Therefore, a penalty should be included. The penalty calculation is similar to the formula of categorical or numerical predicates. Then by applying the same technique as in Section 6.1 and 6.2, the dissimilarity score of Obligation Sets is defined as below.

$$D_{OS}^{\langle Effect\rangle}(P_1,P_2)=\begin{cases}\dfrac{\sum\limits_{(a_{1k},a_{2l})\in\mathcal{M}_a} D_{os}(a_{1k},a_{2l})}{max(N_{a_1},N_{a_2})}, & N_{a_1}>0 \text{ and } N_{a_2}>0\ ;\\ 1, & \text{otherwise}\ .\end{cases} \tag{41}$$

Because the names of attributes are exactly matched, so the dissimilarity score of attribute names is 0. Also, it is important to note that only equality (=) operator is used to compare *AttributeIds* and the score, 0, is assigned to signified the dissimilarity of operators. Therefore, the similarity score of *AttributeIds* are defined as below.

$$\begin{aligned} D_{os}(a_1,a_2) &= \tfrac{1}{3}\left[0+\frac{\sum\limits_{(v_{1k},v_{2l})\in\mathcal{M}_v} d_{os-val}(v_{1k},v_{2l})+\theta_{os-val}}{max(N_{v_1},N_{v_2})}+0\right]\\ &= \tfrac{1}{3}\left[\frac{\sum\limits_{(v_{1k},v_{2l})\in\mathcal{M}_v} d_{os-val}(v_{1k},v_{2l})+\theta_{os-val}}{max(N_{v_1},N_{v_2})}+0\right]. \end{aligned} \tag{42}$$

$$\theta_{os}=\begin{cases}\dfrac{\sum\limits_{(v_{1k},-)\notin\mathcal{M}_v}\sum\limits_{l=1}^{N_{v_2}} d_v(v_{1k},v_{2l})}{N_{v_2}}, & N_{v_1}>N_{v_2}\ ;\\ \dfrac{\sum\limits_{(-,v_{2l})\notin\mathcal{M}_v}\sum\limits_{k=1}^{N_{v_1}} d_v(v_{1k},v_{2l})}{N_{v_1}}, & N_{v_1}<N_{v_2}\ .\end{cases} \tag{43}$$

Because the value is text, therefore, the operator for string data type should be applied, namely, *string-equal, string-greater-than, string-greater-than-or-equal, string-less-than* and *string-less-than-or-equal*. A score of 1 should be applied for any of these operators.

$$d_{os-val}(v_1,v_2)=\begin{cases}0, & \text{if } \textit{string-equal} \text{ is applied}\ ;\\ 1, & \text{otherwise}\ .\end{cases} \tag{44}$$

7.4. Attribute-based Dissimilarity Score of Rule Sets

Similar to the similarity score, in order to calculate the attribute-based dissimilarity score of Rules in the Rule Set, the whole set will be divided into two subsets namely Permit Rule Set and Deny Rule Set based on the Effect of the Rules (Permit/Deny). Each single rule in each policy is then compared with a rule in another policy that has the different effect, and a dissimilarity score of two rules is obtained. Based on the definition of attribute-based dissimilarity in Section 2.1, the idea is to find the rules contain same effect but have the different attributes. In other words, it is the target to find rules that have the potential to yield different results by having different attributes. Because of the mapping of same effects, the Φ mapping and the original procedure to compute the Φ mapping can be reused (mapping of $PR_1(PR_2)$ and $DR_1(DR_2)$). Lin et al.'s original procedure is illustrated in Algorithm A.2 (Appendix A).

The dissimilarity score obtained between the rules is then used to find one-to-many mappings (Φ mappings) for each rule in the two policies. To improve the efficiency, the mapping is only formed if the relatedness score of rules are higher than the threshold value (σ^A). For each rule in P_1, the Φ mappings give similar rules in P_2 which satisfy certain dissimilarity threshold and vice versa. The attribute-based dissimilarity score is a value between 0 and 1, which reflects how dissimilar these rules are with respect to the targets

they are applicable to, the attribute they possess and also with respect to the conditions they impose on the requests.

By using the Φ mapping, the attribute-based dissimilarity score of a rule and a policy can be computed via how dissimilar a rule is with respect to the entire policy by comparing the single rule in one policy with a set of similar rules in the other policy. The notation $rd^A_{1i}(rd^A_{2j})$ denotes the score of a rule $r_{1i}(r_{2j})$ in policy $P_1(P_2)$. The rule attribute-based dissimilarity score $rd_{1i}(rd_{2j})$ is the average of the scores of a rule $r_{1i}(r_{2j})$ and the rules dissimilar to it given by the Φ mapping.

The score of the Rule Set D^A_{RS} are calculated by dividing into two sub-scores namely D^P_{RS} (score of rules with permit-permit effect mapping - Φ^P_1 and Φ^P_2) and D^D_{RS} (score of rules with deny-deny effect mappings - Φ^D_1 and Φ^D_2).

$$\begin{gathered} D^A_{RS} = w^A_P D^P_{RS} + w^A_D D^D_{RS} \ ; \\ \text{where } w^A_P + w^A_D = 1 \ . \end{gathered} \tag{45}$$

w^A_P and w^A_D are weight factors for Permit and Deny rule sets. Then, by using the Φ mapping and calculating the one-to-many mapping for each permit and deny rule, D^P_{RS} and D^D_{RS} are defined in the below equations.

$$D^P_{RS} = \frac{\sum_{i=1}^{N_{PR_1}} rd^A_1 i + \sum_{j=1}^{N_{DR_2}} rd^A_2 j}{N_{PR_1} + N_{PR_2}} \tag{46}$$

$$D^D_{RS} = \frac{\sum_{i=1}^{N_{DR_1}} rd^A_1 i + \sum_{j=1}^{N_{PR_2}} rd^A_2 j}{N_{DR_1} + N_{DR_2}} \ . \tag{47}$$

$$rd^A_{1i} = \begin{cases} \dfrac{\sum_{r_j \in \Phi^P_1(r_{1i})} D^A_{rule}(r_{1i}, r_j)}{|\Phi^P_1(r_{1i})|}, & r_{1i} \in PR_1 \ ; \\ \dfrac{\sum_{r_j \in \Phi^D_1(r_{1i})} D^A_{rule}(r_{1i}, r_j)}{|\Phi^D_1(r_{1i})|}, & r_{1i} \in DR_1 \ . \end{cases} \tag{48}$$

$$rd^A_{2j} = \begin{cases} \dfrac{\sum_{r_i \in \Phi^P_2(r_{2j})} D^A_{rule}(r_{2j}, r_i)}{|\Phi^P_2(r_{2j})|}, & r_{2j} \in PR_2 \ ; \\ \dfrac{\sum_{r_i \in \Phi^D_2(r_{2j})} D^A_{rule}(r_{2j}, r_i)}{|\Phi^D_2(r_{2j})|}, & r_{2j} \in DR_2 \ . \end{cases} \tag{49}$$

D^A_{rule} is the attribute-based dissimilarity score of each pair of rules as the result of the Φ mapping. Similar to the algorithm to compute similarity score, the procedure for computing rule dissimilarity is based on the idea of summing all dissimilar policies in the Φ mapping.

7.5. Attribute-based Dissimilarity Score of Rules

This is the step to calculate dissimilarity scores for each individual pair of rules in the Φ mapping. It is fairly simple because the total score is a sum of attribute-based dissimilarity scores of Rules Targets and Conditions. It is important to note that because the rule's Effect is already utilised to divide the whole Rule Set into Permit and Deny subsets, it should be not counted again in this step. As a rule has Targets and Conditions and a rule's Target consists of Subjects, Resources and Actions, the formulae to calculate dissimilarity scores D^A_{rule} for a pair of rules. Unlike the original algorithm, it is important to note that the rule's Target element should have a higher weight factor than other elements. Similar to the policy's Target element, a checking against the threshold value should be included to improve the efficiency of the overall algorithm.

$$D^A_{rule}(r_i, r_j) = \begin{cases} w^A_d D^A_t(r_i, r_j) + w^A_c D^A_c(r_i, r_j), & R_{rule} \geq \sigma^A \ ; \\ 0, & \text{otherwise} \ . \end{cases} \quad \text{where } w^A_t + w^A_c = 1 \ . \tag{50}$$

$$D^A_t(r_i, r_j) = w^A_s D^A_s(r_i, r_j) + w^A_r D^A_r(r_i, r_j) + w^A_a D^A_a(r_i, r_j) \ ; \quad \text{where } w^A_s + w^A_r + w^A_a = 1 \ . \tag{51}$$

In the above formulae, w^A_t, w^A_c, w^A_s, w^A_r and w^A_a are the weight factors with of the corresponding elements. As Target, Condition, Subject, Resource and Action consist of either categorical or numerical predicates, the same formulae and techniques can be applied to calculate attributed dissimilarity scores.

7.6. Overall Algorithm

The modified algorithm for Policy Dissimilarity Measure is presented in Algorithm 7.1. Lines 1 to 8 are the additional steps to take into account the Target, Rule-Combining Algorithm and Obligation Set element. Line 2 is specifically included to reduce the processing time by checking the relatedness score of Policy against the threshold value. Line 9 categorises rules in P_1 and P_2 based on their effects. Lines 10 to 15 compute the score D^A_{rule} for each pair of rules in P1 and P2. Lines 16 to 19 compute the Φ mappings. Lines 20 to 34 use the Φ mappings to calculate the dissimilarity scores of Rule Sets and sum them up to achieve the scores D^A_{RS} for Rule Sets. Finally, line 35 calculates the overall attribute-based dissimilarity score by adding the scores of Target, Rule-Combining algorithm, Obligation Set and Rule Set.

8. Effect-based Dissimilarity Score of Policy or Rule Elements

As defined in Section 2.1, effect-based dissimilarity takes into account policies with similar attributes but likely to generate different results. Therefore, the approach to compute effect-based dissimilarity score is based on the ***similarity scores*** of the involved elements. Thus, the formulae in Section 4 will be re-used to determine the score of rule elements. The effect-based dissimilarity score of element, categorical predicates and numerical predicates are defined as below.

$$D^E_{\langle Element \rangle}(r_i, r_j) = S_{\langle Element \rangle}(r_i, r_j) \ . \tag{52}$$

9. Policy Effect-based Dissimilarity Measure

The overall effect-based dissimilarity score of policies D^E_{policy} is defined as below.

$$D^E_{policy}(P_1, P_2) = \begin{cases} w^E_T D^E_T(P_1, P_2) + w^E_{OS} D^E_{OS} + w^E_{RS} D^E_{RS}(P_1, P_2), & R_{policy} \geq \sigma^E \ ; \\ 1, & \text{otherwise} \ . \end{cases} \quad \text{where } w^E_T + w^E_{OS} + w^E_{RS} = 1 \ . \tag{53}$$

As mentioned above, the formulae to compute effect-based dissimilarity score of Target and Obligation Set are same as the formulae to computing similarity score. Therefore, the effect-based dissimilarity score of Targets (D^E_T) and Obligation Sets (D^E_{OS}) are defined as below.

$$D^E_T = S_T \ . \tag{54}$$

$$D^E_{OS} = S_{OS} \ . \tag{55}$$

It is also important to note that, similar to Section 5, to maintain the safety nature, Rule-Combining Algorithm will not be considered. Also, the score for Rule Sets and Rules, however, are defined differently due to the different mapping approach. In order to calculate the effect-based dissimilarity score of Rules in the Rule Set, *the only difference is the way in which rule mappings are created.*

Algorithm 7.1: $PolicyAttribute-basedDissimilarityMeasure(P_1, P_2)$ - The Algorithm for Policy Attribute-based Dissimilarity Measure.

input : P_1 with n rules $\{r_{11}, r_{12}, ..., r_{1n}\}$; P_2 with m rules $\{r_{21}, r_{22}, ..., r_{2m}\}$.

output: Policy attribute-based dissimilarity score $D^A_{policy}(P_1, P_2)$.

/* Compute Relatedness Score */

$R_{policy}(P_1, P_2)$

/* Comparing the Relatedness Score with threshold */

if $R_{policy} < \sigma^A$ **then**

  $D^A_{policy}(P_1, P_2) = 1$

  return $D^A_{policy}(P_1, P_2)$

end

/* Compute Score of Combining Algorithms */

$D^A_{CA}(P_1, P_2)$

/* Compute Score of Obligation Sets */

Categorise Obligation Sets based on $FulfillOn$ value (Permit/Deny).

$D^A_{OS}(P_1, P_2)$

Categorise rules in P_1 and P_2 based on their effects. Let $PR_1(PR_2)$ and $DR_1(DR_2)$ denote the set of permit and deny rules respectively in $P_1(P_2)$.

/* Scores for each rule in P_1 and P_2 */

foreach *rule* $r_{1i} \in PR_1$ **do**

  foreach *rule* $r_{2j} \in PR_2$ **do** $D^A_{rule}(r_{1i}, r_{2j})$

end

foreach *rule* $r_{1i} \in DR_1$ **do**

  foreach *rule* $r_{2j} \in DR_2$ **do** $D^A_{rule}(r_{1i}, r_{2j})$

end

/* Compute Φ mappings */

$\Phi^P_1 \leftarrow ComputePhiMapping(PR_1, PR_2, \sigma^A)$

$\Phi^D_1 \leftarrow ComputePhiMapping(DR_1, DR_2, \sigma^A)$

$\Phi^P_2 \leftarrow ComputePhiMapping(PR_2, PR_1, \sigma^A)$

$\Phi^D_2 \leftarrow ComputePhiMapping(DR_2, DR_1, \sigma^A)$

/* Compute the Rule Set dissimilarity scores */

foreach *rule* $r_{1i} \in P_1$ **do**

  if $r_{1i} \in PR_1$ **then**

    $rd^A_{1i} \leftarrow ComputeRuleAttribute-basedDissimilarity(r_{1i}, \Phi^P_1)$

  else if $r_{1i} \in DR_1$ **then**

    $rd^A_{1i} \leftarrow ComputeRuleAttribute-basedDissimilarity(r_{1i}, \Phi^D_1)$

end

foreach *rule* $r_{2j} \in P_2$ **do**

  if $r_{2j} \in PR_2$ **then**

    $rd^A_{2j} \leftarrow ComputeRuleAttribute-basedDissimilarity(r_{2j}, \Phi^P_2)$

  else if $r_{1i} \in DR_2$ **then**

    $rd^A_{2j} \leftarrow ComputeRuleAttribute-basedDissimilarity(r_{2j}, \Phi^D_2)$

end

$D^P_{RS} \leftarrow$ average of rd of permit rules

$D^D_{RS} \leftarrow$ average of rd of deny rules

$D^A_{RS} = w^A_P D^P_{RS} + w^A_D S^D_{RS}$

/* Compute the overall score */

$D^A_{policy}(P_1, P_2) = w^A_T S^A_T(P_1, P_2) + w^A_{CA} S^A_{CA} + w^A_{OS} S^A_{OS} + w^A_{RS} D^A_{RS}(P_1, P_2)$

return $D^A_{policy}(P_1, P_2)$

Algorithm 7.2: *ComputeRuleAttribute − basedDissimilarity*(r', Φ) Algorithm - The Procedure for Computing Rule Attribute-based Dissimilarity.

input : r' is a rule and Φ is a mapping between rules.

output: Rule attribute-based dissimilarity score rd^A.

foreach *rule* $r'' \in \Phi$ **do**
    $sum = sum + D^A_{rule}(r', r'')$
end
$rd^A = \frac{sum}{|\Phi|}$
return rd^A

9.1. Effect-based Dissimilarity Score of Rule Sets

Specifically, the whole set will be divided into two subsets namely Permit Rule Set and Deny Rule Set based on the Effect of the Rules (Permit/Deny). Each single rule in each policy is then compared with a rule in another policy that has the different effect, and a dissimilarity score of two rules is obtained. The idea is to find the rules that contain similar attributes but have the contradictory effect. In other words, it is the target to find rules that have the potential to yield different results with the same attributes.

The effect-based dissimilarity score obtained between the rules is then used to find one-to-many mappings (Ω mappings) for each rule in the two policies. There are four mappings to consider namely $PR_1(DR_2)$ - Ω^P_1, $DR_1(PR_2)$ - Ω^D_1, $PR_2(DR_1)$ - Ω^P_2 and $DR_2(PR_1)$ - Ω^D_2. For example, the mapping $PR_1(PR_2)$ maps each Permit Rule r_{1i} in P_1 with one or more Deny Rules r_{2j} in P_2. Similarly, the mapping $PR_2(DR_1)$ maps each Permit Rule r_{2j} in P_2 with one or more Permit Rules r_{1i} in P_1. To improve the efficiency, the mapping is only formed if the relatedness score of rules are higher the threshold value (σ^E). For each rule in P_1, the Ω mappings give similar rules in P_2 which satisfy certain dissimilarity threshold and vice versa. The Ω mapping is organised in the ways that rules with same attributes but different Effect element should be mapped together to reflect the interpretation of dissimilarity as discussed on Section 2.1. The effect-based dissimilarity score is a value between 0 and 1, which reflects how dissimilar these rules are with respect to the targets they are applicable to, the effect they possess and also with respect to the conditions they impose on the requests.

Algorithm 9.1: *ComputeOmegaMapping*(R', R'', σ^E) Algorithm - The Procedure for Computing Ω Mapping.

input : R' and R'' are sets of rules and σ^E is a threshold value.

output: Ω mapping.

foreach *rule* $r' \in R'$ **do**
    $\Omega(r') = \emptyset$
    foreach *rule* $r'' \in R''$ **do**
        if $S_{rule}(r', r'') \geq \sigma^E$ **then**
            $\Omega(r') = \Omega(r') \cup \{r''\}$
        end
    end
end
return Ω

By using the Ω mappings, the effect-based dissimilarity score of a rule and a policy can be computed via how dissimilar a rule is with respect to the entire policy by comparing the single rule in one policy with a set

of similar rules in the other policy. The notation $rd_{1i}(rd_{2j})$ denotes the dissimilarity score of a rule $r_{1i}(r_{2j})$ in policy $P_1(P_2)$. The rule dissimilarity score $rd_{1i}(rd_{2j})$ is the average of the dissimilarity scores of a rule $r_{1i}(r_{2j})$ and the rules dissimilar to it given by the Ω mapping.

The effect-based dissimilarity score of the Rule Set D^E_{RS} are calculated by dividing into two sub-scores namely D^P_{RS} (score of rules with permit-deny effect mappings - Ω^P_1 and Ω^P_2) and D^D_{RS} (score of rules with deny-permit effect mappings - Ω^D_1 and Ω^D_2).

$$\begin{gathered} D^E_{RS} = w^E_P D^P_{RS} + w^E_D D^D_{RS} \ ; \\ \text{where } w^E_P + w^E_D = 1 \ . \end{gathered} \tag{56}$$

w^E_P and w^E_D are weight factors for Permit and Deny rule sets. By using the Ω mapping and calculating the one-to-many mapping for each permit and deny rule, D^P_{RS} and D^D_{RS} are defined in the following equations:

$$D^P_{RS} = \frac{\sum_{i=1}^{N_{PR_1}} rd_1 i + \sum_{j=1}^{N_{DR_2}} rd_2 j}{N_{PR_1} + N_{DR_2}} \ . \tag{57}$$

$$D^D_{RS} = \frac{\sum_{i=1}^{N_{DR_1}} rd_1 i + \sum_{j=1}^{N_{PR_2}} rd_2 j}{N_{DR_1} + N_{PR_2}} \ . \tag{58}$$

$$rd^E_{1i} = \begin{cases} \dfrac{\sum_{r_j \in \Omega^P_1(r_{1i})} D^E_{rule}(r_{1i}, r_j)}{|\Omega^P_1(r_{1i})|}, & r_{1i} \in PR_1 \ ; \\ \dfrac{\sum_{r_j \in \Omega^D_1(r_{1i})} D^E_{rule}(r_{1i}, r_j)}{|\Omega^D_1(r_{1i})|}, & r_{1i} \in DR_1 \ . \end{cases} \tag{59}$$

$$rd^E_{2j} = \begin{cases} \dfrac{\sum_{r_i \in \Omega^P_2(r_{2j})} D^E_{rule}(r_{2j}, r_i)}{|\Omega^P_2(r_{2j})|}, & r_{2j} \in PR_2 \ ; \\ \dfrac{\sum_{r_i \in \Omega^D_2(r_{2j})} D^E_{rule}(r_{2j}, r_i)}{|\Omega^D_2(r_{2j})|}, & r_{2j} \in DR_2 \ . \end{cases} \tag{60}$$

D^E_{rule} is the effect-based dissimilarity score of each pair of rules as the result of the Ω mapping. The procedure for computing effect-based dissimilarity of rules is based on the idea of evaluating all effect-based dissimilar policies in the Ω mapping.

9.2. Effect-based Dissimilarity Score of Rules

This is the step to calculate dissimilarity scores for each individual pair of rules in the Ω mapping. It is fairly simple because the total score is a sum of dissimilarity scores of Rules Targets and Conditions. It is important to note that because the rule's Effect is already utilised to divide the whole Rule Set into Permit and Deny subsets, it should be not counted again in this step. Unlike the original algorithm, it is important to note that the rule's Target element should have higher weight factor than other elements. Similar to the policy's Target element, a checking against the threshold value should be included to improve the efficiency of the overall algorithm. As a rule has Targets and Conditions and a rule's Target consists of Subjects, Resources and Actions, the formulae to calculate dissimilarity scores D^E_{rule} for a pair of rules are defined as follows.

$$\begin{gathered} D^E_{rule}(r_i, r_j) = \begin{cases} w^E_d D_t(r_i, r_j) + w^E_c D_c(r_i, r_j), & R_{rule} \geq \sigma^E \ ; \\ 0, & \text{otherwise} \ . \end{cases} \\ \text{where } w^E_t + w^E_c = 1 \ . \end{gathered} \tag{61}$$

$$\begin{gathered} D^E_t(r_i, r_j) = w^E_s D^E_s(r_i, r_j) + w^E_r D^E_r(r_i, r_j) + w^E_a D^E_a(r_i, r_j) \ ; \\ \text{where } w^E_s + w^E_r + w^E_a = 1 \ . \end{gathered} \tag{62}$$

In the above formulae, w_t^E, w_c^E, w_s^E, w_r^E and w_a^E are the weight factors of the corresponding elements. As Target, Condition, Subject, Resource and Action consist of either categorical or numerical predicates, the same formulae and techniques can be applied to calculate dissimilarity scores.

9.3. Overall Algorithm

The modified or new computational steps are presented in bold. The modified algorithm for Policy Dissimilarity Measure is presented in Algorithm 9.2. Lines 1 to 7 are the additional steps to take into account the Target, Rule-Combining Algorithm and Obligation Set element. Line 2 is specifically included to reduce the processing time by checking the relatedness score of Policy against the threshold value. Line 8 categorises rules in P_1 and P_2 based on their effects. Lines 9 to 14 compute the dissimilarity score D_{rule} for each pair of rules in P1 and P2. Lines 15 to 18 compute the Ω mappings. Lines 19 to 33 use the Ω mappings to calculate the dissimilarity scores of Rule Sets and sum them up to achieve the dissimilarity scores D_{RS}^E for Rule Sets. Finally, line 34 calculates the overall dissimilarity score by adding the scores of Target, Rule-Combining algorithm, Obligation Set and Rule Set.

10. A Unified Algorithm

Due to the similarity in mapping patterns, it is possible to combine all three algorithm of computing similarity, effect-based dissimilarity and attribute-based dissimilarity into one unified algorithm. The idea is that it is necessary to compute two different types of mappings (Φ and Ω) (Algorithm 10.1). It is important to note that the overall complexity does not increase because the new unified algorithm and the individual algorithms are sharing the same mapping approach. Therefore, similar to the individual algorithms, the unified algorithm does not compromise the light-weight nature of the original approach of Lin et al.

11. Case Study and Discussion

11.1. Case Study

This section presents how the approach can be used to identify similar/dissimilar policies in the three scenarios discussed on Section 2.2. It should be noted that, for the purpose of demonstration, the policies in this section are written in a non-normative form using XACML syntax. These policies are not meant to be processable by XACML processing engines. In addition, it is also assumed that the involved agencies in this case study have a common hierarchy (Figure 5).

Assume that currently, there are three systems in the federation, namely QPIF, ES and QH for a joint inter-agency research project (Scenario 1). A new domain - TMS wants to join the federation. As mentioned in the Scenario 1, TMS must supply involved policies for evaluation. It is supposed that all current domains in the federation have already achieved the common security context and the context is governed by policies that are distributed with member systems.

To simplify the scenario, it is provided that TMS has only one policy that should be considered for evaluation. In this scenario, assume P_1 is the common security policy of the federation that governs accesses to the collaboration resources. P_1 allows research staff and technical support staff access to resources that have the size less than 100MB. The obligation is that upon a successful access, an email will be generated and send to the resource owner. P_1 denies write operation for post-docs, students and technical staff between 19:00 and 21:00. P_2 and P_3 are the policies of TMS. P_2 allows students, research staff and technical staffs access to resources with file size lass than 120MB and without time constraint. P_2 specifically permits technical staff access to server from 19:00 to 22:00 for backing up and denies students' write operations at that time. P_2 denies all requests to access media files. P_3 reflects the policy of TMS that allows certain administrative staff that facilitate the external collaboration activities by using the QPIF mail server from 8:00 to 17:00. Before further and comprehensive analysis, the two policies must be evaluated to see if the policy of TMS is related and similar enough for any further consideration.

The similarity score of P_1 and P_2 is calculated as below with the threshold of 0.5. Since this is the filtering stage, the threshold is set relatively low to include as many potentially similar policies as possible. By applying the formulae in Section 4 and 5, the similarity scores can be calculated as below.

1. Calculate similarity scores for Target, Combining Algorithm, and Obligation Set: $S_T = 0.83$ and $S_{OS} = 0$.

Algorithm 9.2: $PolicyEffect-basedDissimilarityMeasure(P_1, P_2)$ Algorithm - The Algorithm for Policy Effect-based Dissimilarity Measure.

input : P_1 with n rules $\{r_{11}, r_{12}, ..., r_{1n}\}$; P_2 with m rules $\{r_{21}, r_{22}, ..., r_{2m}\}$.

output: Policy dissimilarity score $D^E_{policy}(P_1, P_2)$.

`/* Compute Relatedness Score */`

$R_{policy}(P_1, P_2)$

`/* Comparing the Relatedness Score with threshold */`

if $R_{policy} < \sigma^E$ **then**

  $D^E_{policy}(P_1, P_2) = 1$

  return $D^E_{policy}(P_1, P_2)$

end

`/* Scores of Obligation Sets */`

Categorise Obligation Sets based on $FulfillOn$ value (Permit/Deny).

$D^E_{OS}(P_1, P_2)$

Categorise rules in P_1 and P_2 based on their effects. Let $PR_1(DR_2)$ and $DR_1(PR_2)$ denote the set of permit and deny rules respectively in $P_1(P_2)$.

`/* Scores for each rule in` P_1 `and` P_2 `*/`

foreach *rule* $r_{1i} \in PR_1$ **do**

  foreach *rule* $r_{2j} \in DR_2$ **do** $D^E_{rule}(r_{1i}, r_{2j})$

end

foreach *rule* $r_{1i} \in DR_1$ **do**

  foreach $r_{2j} \in PR_2$ **do** $D^E_{rule}(r_{1i}, r_{2j})$

end

`/* Compute Ω mappings */`

$\Omega^P_1 \leftarrow ComputeOmegaMapping(PR_1, DR_2, \sigma^E)$

$\Omega^D_1 \leftarrow ComputeOmegaMapping(DR_1, PR_2, \sigma^E)$

$\Omega^P_2 \leftarrow ComputeOmegaMapping(PR_2, DR_1, \sigma^E)$

$\Omega^D_2 \leftarrow ComputeOmegaMapping(DR_2, PR_1, \sigma^E)$

`/* Compute the Rule Set dissimilarity scores */`

foreach *rule* $r_{1i} \in P_1$ **do**

  if $r_{1i} \in PR_1$ **then**

    $rd^E_{1i} \leftarrow ComputeRuleEffect-basedDissimilarity(r_{1i}, \Omega^P_1)$

  else if $r_{1i} \in DR_1$ **then**

    $rd^E_{1i} \leftarrow ComputeRuleEffect-basedDissimilarity(r_{1i}, \Omega^D_1)$

end

foreach *rule* $r_{2j} \in P_2$ **do**

  if $r_{2j} \in PR_2$ **then**

    $rd^E_{2j} \leftarrow ComputeRuleEffect-basedDissimilarity(r_{2j}, \Omega^P_2)$

  else if $r_{1i} \in DR_2$ **then**

    $rd^E_{2j} \leftarrow ComputeRuleEffect-basedDissimilarity(r_{2j}, \Omega^D_2)$

end

$D^P_{RS} \leftarrow$ average of rd of permit-deny rules

$D^D_{RS} \leftarrow$ average of rd of deny-permit rules

$D^E_{RS} = w^E_P S^P_{RS} + w^E_D S^D_{RS}$

`/* Compute the overall score */`

$D^E_{policy}(P_1, P_2) = w^E_T D^E_T(P_1, P_2) + w^E_{OS} D^E_{OS} + w^E_{RS} D^E_{RS}(P_1, P_2)$

return $D^E_{policy}(P_1, P_2)$

Algorithm 9.3: $ComputeRuleEffect-basedDissimilarity(r', \Omega)$ Algorithm - The Procedure for Computing Rule Effect-based Dissimilarity.

input : r' is a rule and Ω is a mapping between rules.

output: Rule effect-based dissimilarity score rd^E.

foreach *rule* $r'' \in \Omega$ **do**
    $sum = sum + D^E_{rule}(r', r'')$
end
$rd^E = \frac{sum}{|\Omega|}$
return rd^E

Algorithm 10.1: $UnifiedMeasure(P_1, P_2)$ Algorithm - The Unified Algorithm..

input : P_1 with n rules $\{r_{11}, r_{12}, ..., r_{1n}\}$; P_2 with m rules $\{r_{21}, r_{22}, ..., r_{2m}\}$.

output: Similarity or Dissimilarity score.

`/* Compute Relatedness Score */`
$R(P_1, P_2)$
Comparing the Relatedness Score with appropriate threshold
Categorise Obligation Sets based on $FulfillOn$ value (Permit/Deny)
$S_{OS}(P_1, P_2)$
$D^E_{OS}(P_1, P_2)$
Categorise rules in P_1 and P_2 based on their effects
Compute similarity/dissimilarity score for each rule
Compute S_{rule}, D^A_{rule} and D^E_{rule}
Φ Mapping $\leftarrow ComputePhiMapping$
Ω Mapping $\leftarrow ComputeOmegaMapping$
Compute Rule Similarity (rs) using Φ mappings $\leftarrow ComputeRuleSimilarity$
Compute Rule Attribute-based Dissimilarity (rd^A) using Φ mappings $\leftarrow ComputeRuleAttribute-basedDissimilarity$
Compute Rule Effect-based Dissimilarity (rd^E) using Ω mappings $\leftarrow ComputeRuleEffect-basedDissimilarity$
Compute S_{RS}, D^A_{RS}, D^E_{RS}
Compute S_{policy}
Compute D^A_{policy} and D^E_{policy}

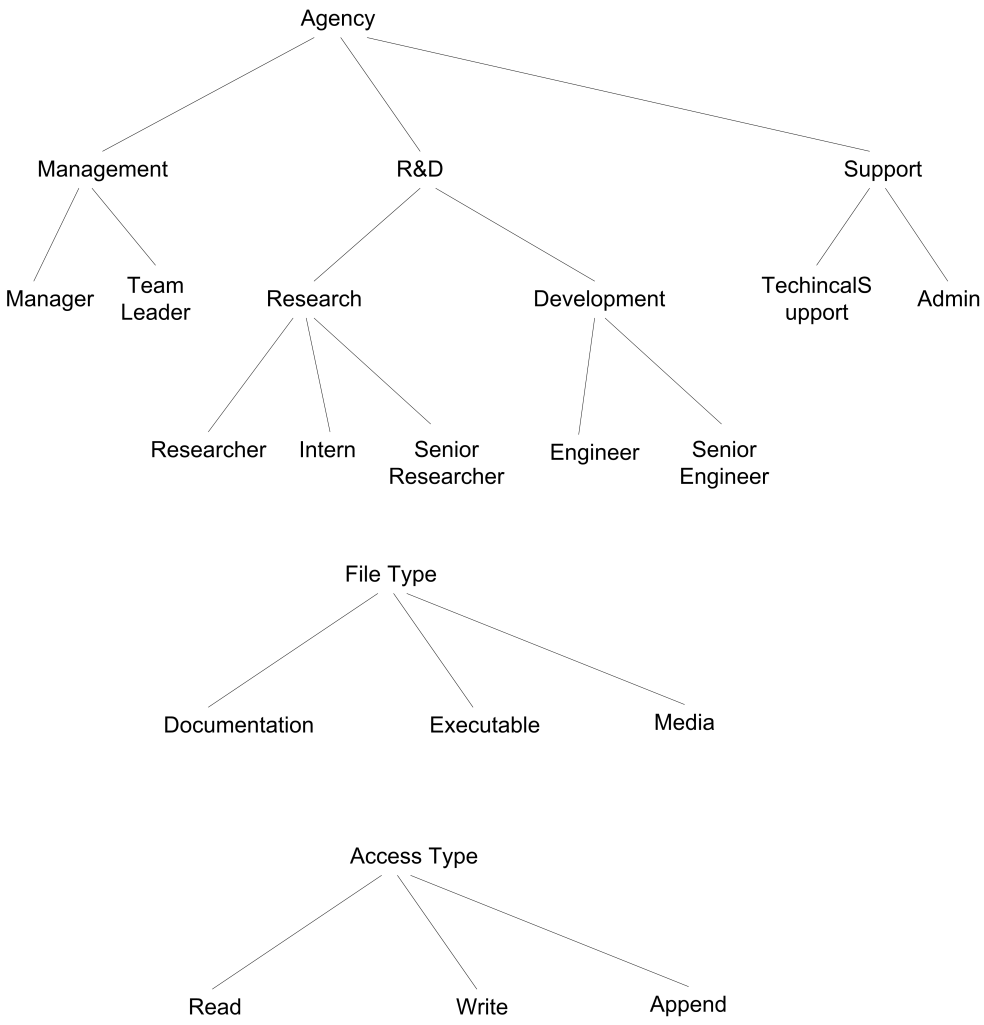

Figure 5: The Common Hierarchy for the Case Study.

2. Categorise rules in P_1 and P_2 based on their effects and find the permit and deny rule sets, $PR_1(PR_2)$ and $DR_1(DR_2)$. These sets are $PR1 = \{R_{11}\}$, $PR_2 = \{R_{21}, R_{22}\}$, $DR_1 = \{R12\}$, $DR_2 = \{R_{23}, R_{24}\}$.

3. Rule similarity scores between pairs of rules with the same effect in both policies are:

$$S_{rule}(R_{11}, R_{21}) = 0.79 .$$
$$S_{rule}(R_{11}, R_{22}) = 0.63 .$$
$$S_{rule}(R_{12}, R_{23}) = 0.81 .$$
$$S_{rule}(R_{12}, R_{24}) = 0.42 .$$

4. Compute Φ mappings using threshold value of 0.5:

$$\Phi_1^P = \{R_{11} \rightarrow \{R_{21}, R_{22}\}\} .$$
$$\Phi_1^D = \{R_{12} \rightarrow \{R_{23}\}\} .$$
$$\Phi_2^P = \{R_{21} \rightarrow \{R_{11}\}, R_{22} \rightarrow \{R_{11}\}\} .$$
$$\Phi_2^D = \{R_{23} \rightarrow \{R_{12}\}\} .$$

5. For each rule r_{1i} in P_1, the corresponding rule similarity score rs_{1i} is computed:

$$rs_{11} = \tfrac{1}{2}\{S_{rule}(R_{11}, R_{21}) + S_{rule}(R_{11}, R_{22})\} = 0.71 .$$
$$rs_{12} = S_{rule}(R_{12}, R_{23}) = 0.81 .$$

6. For each rule r_{2j} in P_2, the corresponding rule similarity score rs_{2j} is computed:

$$rs_{21} = S_{rule}(R_{11}, R_{21}) = 0.79 .$$
$$rs_{22} = S_{rule}(R_{11}, R_{22}) = 0.63 .$$
$$rs_{23} = S_{rule}(R_{12}, R_{23}) = 0.81 .$$
$$rs_{24} = 0 .$$

7. Then, the similarity between the permit rule sets of P_1 and P_2, given by S_{RS}^P is computed:

$$S_{RS}^P = \frac{rs_{11} + rs_{21} + rs_{22}}{3} = \frac{0.71 + 0.79 + 63}{3} = 0.71 .$$

8. The similarity between the deny rule sets of P_1 and P_2, given by S_{RS}^D is computed:

$$S_{RS}^D = \frac{rs_{12} + rs_{23} + rs_{24}}{3} = \frac{0.81 + 0.81 + 0}{3} = 0.54 .$$

9. Compute the overall similarity score of Rule Sets with weight factors are evenly distributed:

$$S_{RS} = w_P S_{RS}^P + w_D S_{RS}^D = \frac{1}{2}0.50 + \frac{1}{2}0.71 = 0.63 .$$

10. Finally the similarity score of permit/deny rule sets and Policy's Target are combined to obtain the overall policy similarity score of policies P_1 and P_2 which weight factors are unevenly distributed to reflect the importance of the Target and Rule Set:

$S_{policy}(P_1, P_2) =$

$$w_T S_T(P_1, P_2) + w_{OS} S_{OS}(P_1, P_2) + w_{RS} S_{RS} = \frac{3}{8}0.83 + \frac{2}{8}0 + \frac{3}{8}0.63 = 0.55 .$$

In this example, the two compared policies, in general, appear to be similar. However, if taking into account the obligation of P_1, the distinction is quite clear. It is demonstrated that the obligations can play a considerable role in determining the similarity score.

For (P_1, P_3) similarity, by applying the same techniques, the similarity score for Targets is $S_T = 0.33$. Therefore, there is no need to go further as these two policies are unrelated and written for different targets. The similarity score for this pair of policies is 0. This score reflects the different nature of P_1 and P_3 in terms of who should be allowed access the collaborative information. P_1 only allows research and technical staff to get access but P_3 allows administrative staff as well. This difference should be pointed out and P_3 should be returned to TMS and notified to the federation authority for further consideration and negotiation. The results show that the similarity score of P_1 and P_2 is lower than the original algorithm. However, the

similarity score of Rules is not much different. It shows that, in this case, taking into account the nature of the operators does not increase the possibility of ignoring potentially matched rules/policies.

Now, supposed that TMS is qualified for membership of the federation. The collaborative activities require research staff of TMS to access, delegate access or be delegated with access to various involved resources. Now suppose that TMS staff need to get access to a draft of a confidential and patented design of QH. TMS asks for delegation from QH staff. Since the QH is very sensitive about this document, they want to exclude as many staff that can access as possible. Therefore, before any delegation is made, a dynamic evaluation of policies must be made (Scenario 3). Assume that P_4 is the policy that guards the confidential files of the design and P_5 is the policy that TMS suggests to QH that it will apply to guard the resource if the access is granted. P_4 allows research staff and technical staff access the confidential design from 8:00 o 17:00 and specifically denies the access of students and post-docs. P_5 just allows access for research staff with similar time constraint.

By applying the dissimilarity measures in Section 6, 7, 8 and 9, the attribute based dissimilarity score D^A_{policy} is nearly 0 and the effect based dissimilarity score D^E_{policy} is nearly 1.

In this example, instead of focusing on computational and procedural aspect, the example (and the involved policies) are simplified to illustrate the concept of attribute based and effect based dissimilarity. Two policies P_4 and P_5 are largely identical and so the attribute based dissimilarity is not significant. However, the effect can be significantly different. Depending on the objective of the filtering process, appropriate dissimilarity scores can be used. In this example, if the filter aims at detecting any potential conflict, then both scores should be computed. As the dissimilarity scores are high, the policy should be further evaluated.

Via this case study, it can be seen that there are a number of situations in which it is necessary to have a mechanism to identify the potential similar or dissimilar policies. A detailed example is presented to illustrate how the enhanced policy similarity measure algorithm works. For reference purposes, in this paper, similar sample policies from the paper of Lin et al., namely P_1, P_2 and P_3, are reused. The policies are modified to include Rule-Combining Algorithm and Obligation Set element. In this case study, the similarity score of (P_1, P_2) and (P_1, P_3) will be calculated. The threshold values (ϵ and σ) is 0.5. It is important to note that for simplicity, this case study uses a single threshold value and evenly distributed weight factors. In practice, the policy processing engine is free to determine the threshold values and weight factors in any way that serves its purposes. It is important to note that the examples in the case study are adequate to illustrate the involved concepts. Thorough evaluation needs a more complex arrangement and a larger number of policies for processing. It is also important to recognise that the dissimilarity score is not the absolute measure to reject an access request. It is a means to raise alarm about potential conflict of the involved policies. How to deal with the warning is not addressed by this paper. These issues will be discussed further in Section 11.2 as future work.

11.2. Discussion

Consistent with the intent of the original algorithm of Lin et al., the modified algorithm is designed to act as a filter which selects a smaller set of similar or dissimilar policies which can be further evaluated for compatibility or equivalence by using more computationally expensive methods. The modifications still allow the policy processing engine to control the assessed level of similarity of the policies via threshold and weight factors. Therefore, the modifications do not violate the purposes of the original algorithm (as discussed in Section 2.4). In addition, if the policy evaluation process needs to identity not only similar but also dissimilar policies, the sum of the two similar and dissimilar set of policies that satisfy the threshold can be made to form a set of related policies.

In terms of computational complexity, the modifications do not compromise the lightweight nature of the original algorithm. In the enhanced version of similarity evaluation, the most computationally expensive task is also to compute the similarity score S_{RS} of Rule Set and S_{OS} of Obligation Sets. Similar to the original algorithm, if it is supposed that the average number of attributes in one element of rules or obligations is n_a, to find matching attributes with the same name, it takes $O(n_a log(n_a))$ to sort and compare the list of attribute names. As one attribute name is associated with one or very few number of values, it is estimated the time for the attribute value computation to be a constant time c_1. In addition, the *If* statement (line 2 in Figure 5.1) - for checking similarity score of Targets against the threshold value - adds a linear amount of time (c_2) to overall processing time. Therefore, the overall complexity is still $O(n_a log(n_a) + c_1 n_a + c_2)$ which effectively makes it of the same order of complexity as the original algorithm with the average case complexity

of $O(n_a log(n_a))$. Therefore, by taking into account additional factors, the enhanced algorithm can increase the efficiency and reduce the processing time without compromising the complexity of the original algorithm of Lin et al. However, the algorithm may require a longer processing time due to the evaluation of additional factors.

Similar to the similarity algorithm, in the algorithms to evaluate attribute-based dissimilarity and effect-based dissimilarity, the modifications do not compromise the lightweight nature of the original algorithm. The most computationally expensive task is also to compute the dissimilarity score D^A_{RS}/D^E_{RS} of Rule Sets and D^A_{OS}/D^E_{OS} of Obligation Sets. Therefore, the overall average case complexity is also $O(n_a log(n_a))$ which effectively makes it equally complex compared to the original algorithm given that the average number of attributes in one element of rules or obligations is n_a and the If statement (line 2) - for checking relatedness score of Targets against the threshold value - adds a linear amount of time ($c2$) to overall processing time.

By comparing additional elements of policies such as Obligation Set, the enhanced algorithm does not necessarily limit the potentially matched policies. It simply allows the policy processing engine more options when evaluating the similarity. If the current evaluation criteria do not need to include these additional factors, for example as in a discrimination type query [11], it just has to set the respective weight factors to zero.

The algorithm in this paper is designed as an improvement from the original work of Lin et al. However, it still suffers certain limitations which can decrease the efficiency and practicality of the algorithm. Firstly, with the assumption of a common hierarchy, the algorithm only provides a one-to-one mapping between one attribute to another. However, this assumption is not always achievable. Therefore, within the context of attribute evaluation, it is important to consider an appropriate method to provide mapping of different attribute names pointing to the same attributes to support computing the Φ and Ω mappings [15, 17]. In addition, as can be seen, in either the current algorithms or the original algorithm, the way in which weight factors are assigned is implementation-specific. It can lead to the situation in which the same evaluation may be conducted with different weight factors by different security authorities. Therefore, it is important to have a study to achieve a set of thorough criteria for assigning weight factors.

As was previously noted, the algorithms are built on the assumption of the existence of a common hierarchy for policies from different security domains. Therefore, it is also prone to failure if this assumption is not valid. In addition to this, the algorithms are also built based on one-to-one mappings of attributes. It means that if the attributes of the involved policies can not be mapped (for example, similar attributes with different names - not one attribute with many values), the algorithms will fail to produce the evaluation score. Therefore, even if a common hierarchy can not be achieved, it would be useful if there is a mechanism to produce attribute mapping to support the algorithms in this paper. This issue promises potential ground for further work with significant contribution to the development of not only this area of research but also others such as information retrieval or federated database management.

Finally, it may also be interesting to investigate the potential implementation of the policy filtering mechanism with the utilisation of more comprehensive policy evaluation techniques such as SAT solver or MTBDD and investigate the performance of the framework in such conditions. It is also important to note that the current examples are adequate to demonstrate the functionality of the algorithms but they can not be used to thoroughly evaluate the algorithms' performance. Therefore, a more thorough mechanism to evaluate the performance of the algorithms when processing large numbers of policies is necessary. By assessing a large number of policies, it is possible to adjust the algorithms to provide better granularity when comparing policies with marginal similarity/dissimilarity. Finally, it would also be a significant contribution to investigate a mechanism to process the warnings from similar/dissimilar detection to achieve a complete and useful policy conflict prevention system.

12. Summary and Conclusion

This paper proposed a policy filtering algorithm to identify similar/dissimilar policies based on the requirements of the security authorities. A modified algorithm was proposed to evaluate a similarity score of two policies and an algorithm to evaluate a dissimilarity score of two policies based on the relatedness score of two policies to detect potential conflicts. Specifically, a checking mechanism for Target element of Policy and Rule was implemented to further reduce unrelated policies or rules from the whole policy set. Therefore, the algorithm can improve the efficiency and reduce substantially computational efforts without

compromising the light-weight nature of the original approach. By applying the reverse mapping of Permit and Deny Rule Sets combined with an appropriate threshold value, dissimilarity of rules can be achieved. Together with a detailed set of formulas and the algorithm, this paper also applied the enhanced algorithm to a set of policies to illustrate the application of the algorithm.

The main contribution of this paper is the modified algorithms to compute similarity and dissimilarity for policies based on the algorithm of Lin et al [12]. This paper argues that there is a demand for a mechanism to compare and filter policies. This is designed to support the task of comparing policies written in the XACML policy format. The evaluation process is based on two stages. The first stage adopted a light-weight approach to approximately identify as many potentially similar or dissimilar policies as possible. The second stage, which is not within the scope of the paper, employs more computationally expensive analysis approaches using boolean checking or semantic analysis to comprehensively evaluate the policies, which are identified in the first stage. The first stage is designed to bring the high computational workload of the second stage within practical bounds [11].

The work in this paper is an extended version of Lin et al.'s proposal to calculate both the similarity and dissimilarity of policies. The new algorithm also distinguished the concept of relatedness from similarity. In addition to similarity evaluation, dissimilarity scores were also considered. The dissimilarity scores are an important part as they are the indicators for potential conflicts of policies and constraints. The novelty lies in the incorporation of additional factors such as XACML operators and obligations to provide a more efficient evaluation.

References

[1] T. Ahmed and A. R. Tripathi, "Static Verification of Security Requirements in Role Based CSCW Systems", in Proceedings of the 8th ACM Symposium on Access Control Models and Technologies, Como, Italy, 2003, pp. 196-203.

[2] M. Backes, G. Karjoth, W. Bagga, and M. Schunter, "Efficient Comparison of Enterprise Privacy Policies", in Proceedings of the 2004 ACM Symposium on Applied Computing, Nicosia, Cyprus, 2004, pp. 375-382.

[3] A. Budanitsky and G. Hirst, "Evaluating WordNet-based Measures of Lexical Semantic Relatedness", Association for Computational Linguistics, vol. 32(1), 2005.

[4] H. Do and E. Rahm, "COMA: A System for Flexible Combination of Schema Matching Approaches", in Proceedings of the 28th International Conference on Very Large Data Bases (VLDB), Hong Kong, China, 2002, pp. 610-621.

[5] M. Ehrig and S. Staab, "QOM - Quick Ontology Mapping", in Proceedings of the 3rd International Semantic Web Conference (ISWC2004), Hiroshima, Japan, 2004.

[6] J. Euzenat and P. Valtchev, "An Integrative Proximity Measure for Ontology Alignment", in Proceedings of the Workshop on Semantic Information Integration, Sanibel Island, US, 2003, pp. 33-38.

[7] K. Fisler, S. Krishnamurthi, L. A. Meyerovich, and M. C. Tschantz, "Verification and Change-impact Analysis of Access Control Policies", in Proceedings of the 27th International Conference on Software Engineering, St. Louis, MO, USA, 2006, pp. 196-205.

[8] F. Giunchiglia, P. Shvaiko, and M. Yatskevich, "S-Match: an Algorithm and an Implementation of Semantic Matching", in Proceedings of the European Semantic Web Symposium (ESWS) 2004, Heraklio, Germany, 2004, pp. 61-75.

[9] D. P. Guelev, M. Ryan, and P. Y. Schobbens, "Model-Checking Access Control Policies", in Proceedings of the 7th Information Security Conference (ISC), 2004, pp. 219-230.

[10] M. Koch, L. V. Mancini, and F. Parisi-Presicce, "On the Specification and Evolution of Access Control Policies", in Proceedings of the 6th ACM Symposium on Access Control Models and Technologies, Chantilly, Virginia, United States, 2001, pp. 121-130.

[11] D. Lin, P. Rao, E. Bertino, N. Li, and J. Lobo, "EXAM - a Comprehensive Environment for the Analysis of Access Control Policies", CERIAS, Purdue University 2008.

[12] D. Lin, P. Rao, E. Bertino, and J. Lobo, "An Approach to Evaluate Policy Similarity", in Proceedings of the 12th ACM Symposium on Access Control Models and Technologies (SACMAT'07), Sophia Antipolis, France, 2007, pp. 1-10.

[13] A. Malucelli, "Ontology-based Services for Agents Interoperability", PhD Thesis, University of Porto, 2006.

[14] P. Mazzoleni, E. Bertino, B. Crispo, and S. Sivasubramanian, "XACML Policy Integration Algorithms: not to be confused with XACML Policy Combination Algorithms!" in Proceedings of the 11th ACM symposium on Access Control Models and Technologies, Lake Tahoe, California, USA, 2006, pp. 219-227.

[15] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity Flooding: A Versatile Graph Matching Algorithm", in Proceedings of the International Conference on Data Engineering (ICDE), 2002, pp. 117-128.

[16] Organization for the Advancement of Structured Information Standards (OASIS), "eXtensible Access Control Markup Language (XACML) Version 2.0", 2005, Available: `http://www.oasis-open.org/committees/download.php/10577/XACML-2.0-OS-ALL.zip` [10th August 2006]

[17] E. Rahm and P. A. Bernstein, "A Survey of Approaches to Automatic Schema Matching". The International Journal on Very Large Data Bases (VLDB Journal), vol. 10(4), 2005, pp. 334-350.

A. Appendix - Lin et al.'s Original Formulae and Algorithm

The following formulae and procedures present the original approach of Lin et al. to compute similarity score of two policies [12]. For reference purposes, the equations are organised in the same order as in the original paper. Variable names are also kept intact. Original notations are presented in Table 2.

$$rs_{1i} = \begin{cases} \dfrac{\sum\limits_{r_j \in \Phi_1^P(r_{1i})} S_{rule}(r_{1i}, r_j)}{|\Phi_1^P(r_{1i})|}, & r_{1i} \in PR_1 \ ; \\ \dfrac{\sum\limits_{r_j \in \Phi_1^D(r_{1i})} S_{rule}(r_{1i}, r_j)}{|\Phi_1^D(r_{1i})|}, & r_{1i} \in DR_1 \ . \end{cases} \tag{63}$$

$$rs_{2j} = \begin{cases} \dfrac{\sum\limits_{r_i \in \Phi_2^P(r_{2j})} S_{rule}(r_{2j}, r_i)}{|\Phi_2^P(r_{2j})|}, & r_{2j} \in PR_2 \ ; \\ \dfrac{\sum\limits_{r_i \in \Phi_2^D(r_{2j})} S_{rule}(r_{2j}, r_i)}{|\Phi_2^D(r_{2j})|}, & r_{2j} \in DR_2 \ . \end{cases} \tag{64}$$

$$S^P_{rule-set} = \frac{\sum\limits_{i=1}^{N_{PR_1}} rs_1 i + \sum\limits_{j=1}^{N_{PR_2}} rs_2 j}{N_{PR_1} + N_{PR_2}} \ . \tag{65}$$

$$S^D_{rule-set} = \frac{\sum\limits_{i=1}^{N_{DR_1}} rs_1 i + \sum\limits_{j=1}^{N_{DR_2}} rs_2 j}{N_{DR_1} + N_{DR_2}} \ . \tag{66}$$

$$S_{policy}(P_1, P_2) = w_T S_T(P_1, P_2) + w_p S^P_{rule-set} + w_d S^D_{rule-set} \ ; \quad w_T + w_p + w_d = 1 \ .^1 \tag{67}$$

Notation	Meaning
P	Policy
PR	Permit Rule Set
DR	Deny Rule Set
r	Rule
a	Attribute
v	Attribute value
H	Height of a hierarchy
S_{policy}	Similarity score of two policies
S_{rule}	Similarity score of two rules
$S^{P}_{rule-set}$	Similarity score of two permit rule sets
$S^{D}_{rule-set}$	Similarity score of two deny rule sets
$S_{\langle Element \rangle}$	Similarity score of elements, where $\langle Element \rangle \in \{T, t, c, s, r, a\}$
s_{cat}	Similarity score of two categorical values
S_{cat}	Similarity score of two categorical predicates
s_{num}	Similarity score of two numerical values
S_{num}	Similarity score of two numerical predicates
rs	Similarity score of a rule and a policy
Φ	Rule mapping
$\mathcal{M}_a$	Set of pairs of matching attribute names
$\mathcal{M}_v$	Set of pairs of matching attribute values
N_{PR}	Number of permit rules in a policy
N_{DR}	Number of deny rules in a policy
N_a	Number of attributes in an element
N_v	Number of values of an attribute
$SPath$	Length of shortest path of two categorical values
$w_{\langle Element \rangle}$	Weight of similarity score of elements, where $\langle Element \rangle \in \{T, t, c, s, r, a\}$
ϵ	Rule similarity threshold
δ	Compensating score of unmatched values

Table 2: The Original Notations of Lin et al. [12].

$$\Phi(r_i) = \{r_j | S_{rule}(r_i, r_j) \geq \epsilon\} \ . \tag{68}$$

$$\begin{gathered} S_{rule}(r_i, r_j) = w_t S_t(r_i, r_j) + w_c S_c(r_i, r_j) \ ; \\ w_t + w_c = 1 \ . \end{gathered} \tag{69}$$

$$\begin{gathered} S_t(r_i, r_j) = w_s S_s(r_i, r_j) + w_r S_r(r_i, r_j) + w_a S_a(r_i, r_j) \ ; \\ w_s + w_r + w_a = 1 \ . \end{gathered} \tag{70}$$

$$S_{\langle Element \rangle}(r_i, r_j) = \begin{cases} \dfrac{\sum\limits_{(a_{1k}, a_{2l}) \in \mathcal{M}_a} S_{attr-type}(a_{1k}, a_{2l})}{max(N_{a_1}, N_{a_2})}, & N_{a_1} > 0 \text{ and } N_{a_2} > 0 \ ; \\ 1, & \text{otherwise} \ . \end{cases} \tag{71}$$

$$s_{cat}(v_1, v_2) = 1 - \frac{SPath(v_1, v_2)}{2H} \ . \tag{72}$$

$$S_{cat}(a_1, a_2) = \frac{1}{2} \left[1 + \frac{\sum\limits_{(v_{1k}, v_{2l}) \in \mathcal{M}_v} s_{cat}(v_{1k}, v_{2l}) + \delta}{max(N_{v_1}, N_{v_2})} \right] \ . \tag{73}$$

$$\delta = \begin{cases} \dfrac{\sum\limits_{(v_{1k}, -) \notin \mathcal{M}_v} \sum\limits_{l=1}^{N_{v_2}} s_{cat}(v_{1k}, v_{2l})}{N_{v_2}}, & N_{v_1} > N_{v_2} \ ; \\ \dfrac{\sum\limits_{(-, v_{2l}) \notin \mathcal{M}_v} \sum\limits_{k=1}^{N_{v_1}} s_{cat}(v_{1k}, v_{2l})}{N_{v_1}}, & N_{v_1} < N_{v_2} \ . \end{cases} \tag{74}$$

$$s_{num}(v_1, v_2) = \frac{|v_1 - v_2|}{max(v_1, v_2)} \ . \tag{75}$$

$$S_{num}(a_1, a_2) = \frac{1}{2} \left[1 + \frac{\sum\limits_{(v_{1k}, v_{2l}) \in \mathcal{M}_v} s_{num}(v_{1k}, v_{2l}) + \delta}{max(N_{v_1}, N_{v_2})} \right] \ . \tag{76}$$

$$\delta = \begin{cases} \dfrac{\sum\limits_{(v_{1k}, -) \notin \mathcal{M}_v} \sum\limits_{l=1}^{N_{v_2}} s_{num}(v_{1k}, v_{2l})}{N_{v_2}}, & N_{v_1} > N_{v_2} \ ; \\ \dfrac{\sum\limits_{(-, v_{2l}) \notin \mathcal{M}_v} \sum\limits_{k=1}^{N_{v_1}} s_{num}(v_{1k}, v_{2l})}{N_{v_1}}, & N_{v_1} < N_{v_2} \ . \end{cases} \tag{77}$$

In the above formulae, it is important to note that Lin et al. has an interesting approach to compute the scores for categorical ($SPath$ and $Hcode$), numerical and the compensating scores (δ) for unmatched attributes. $SPath$ is an interesting concept which allows the categorical values to be considered not only by exact match but also the semantic similarity [12]. $SPath$ is, in fact, the distance of the shortest path between the two values in a hierarchy tree (Figure 6). Equation 72 presents the formula to compute the

[1] w_p and w_d are the weight factors of similarity scores of Permit Rule Set and Deny Rule Set. They were not described in the original table of notations of Lin et al. (Table 2).

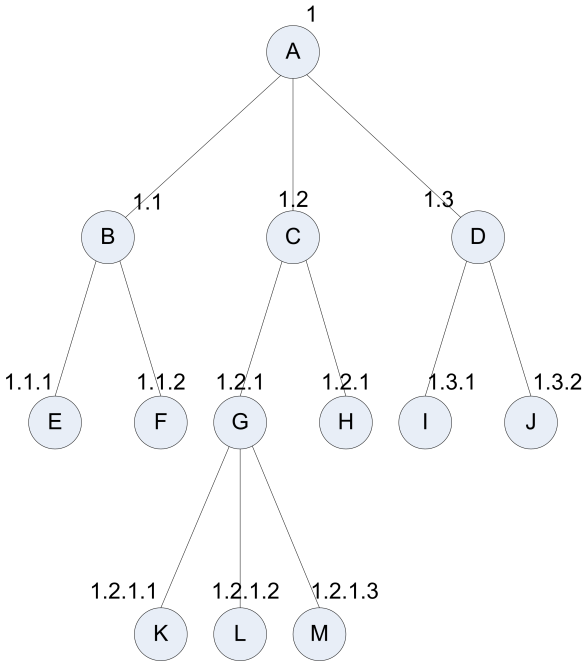

Figure 6: Lin et al.'s Example Hierarchy [12].

similarity score of two categorical values in a hierarchy with a height of H. Therefore, two hierarchically closer values have higher similarity score. In fact, $Hcode$ is a number to indicate the position of nodes in the hierarchy. In this settings, $Hcode$ acts like a cache to increase the efficiency of the algorithm by saving time to traverse along the hierarchy to calculate the $SPath$. By using $Hcode$, $SPath$ can be easily determined by counting the total number of different elements' $Hcode$.

In the original algorithm, the issue of comparing two sets of values is also considered. The original algorithm defined that the similarity score of two sets of values is the sum of scores of pairs $\langle v_{1k}, v_{2l} \rangle$ and the compensating score. This leads to demand for a compensating score (δ). δ is, in fact, the average similarity score of the unmatched pairs. The idea is that if there are some matched predicates, the score should be reduced to reflect the level of similarity (so less dissimilar). To compute the total score of two sets of values, it is important to find $\mathcal{M}_v$ which a set of pairs of values which have the following properties [12]:

1. If $v_{1k} \neq v_{2l}$, then $(v_{1k}, v_{2l}) \in \mathcal{M}_v$.
2. For pairs $v_{1k} = v_{2l}$, pairs contributing to the maximum sum of similarity scores belong to $\mathcal{M}_v$.
3. Each attribute value v_{1k} or v_{2l} occurs at most once in $\mathcal{M}_v$.

It is fairly easy to find pairs that meet the first and the last criterion. However, the second criterion is more difficult. Actually, the second criterion is designed to capture the pairs that can maximise the similarity score. Therefore, the pairs that meet this criterion are the ones have the highest scores among the unmatched values. If there are pairs with equally highest scores, their compensating scores are considered. If the compensating scores are the same, then these pairs are qualified to be included in the $\mathcal{M}_v$.

The original algorithm for Policy Similarity Measure is presented in Algorithm A.2, A.3 and A.1.

B. Appendix - XACML Policies in the Case Study

The following policies are used to illustrate the algorithm in the Case Study. It should be noted that, for the purpose of demonstration, the policies in this section are written in a non-normative form using XACML syntax. These policies are not meant to be processable by XACML processing engines.

Algorithm A.1: $PolicySimilarityMeasure(P_1, P_2)$ Algorithm - Lin et al.'s Algorithm for Policy Similarity Measure [12].

input : P_1 is a policy with n rules $\{r_{11}, r_{12}, ..., r_{1n}\}$ and P_2 is a policy with m rules $\{r_{21}, r_{22}, ..., r_{2m}\}$.

output: Policy similarity score $S_{policy}(P_1, P_2)$.

Categorise rules in P_1 and P_2 based on their effects. Let $PR_1(PR_2)$ and $DR_1(DR_2)$ denote the set of permit and deny rules respectively in $P_1(P_2)$.

`/* Similarity scores for each rule in` P_1 `and` P_2 `*/`
foreach *rule* $r_{1i} \in PR_1$ **do**
  foreach *rule* $r_{2j} \in PR_2$ **do**
    `/* similarity score of rules */`
    $S_{rule}(r_{1i}, r_{2j})$
  end
end
foreach *rule* $r_{1i} \in DR_1$ **do**
  foreach *rule* $r_{2j} \in DR_2$ **do**
    `/* similarity score of rules */`
    $S_{rule}(r_{1i}, r_{2j})$
  end
end
`/* Compute` Φ `mappings */`
$\Phi_1^P \leftarrow \text{ComputePhiMapping}(PR_1, PR_2, \epsilon)$
$\Phi_2^P \leftarrow \text{ComputePhiMapping}(PR_2, PR_1, \epsilon)$
$\Phi_1^D \leftarrow \text{ComputePhiMapping}(DR_1, DR_2, \epsilon)$
$\Phi_2^D \leftarrow \text{ComputePhiMapping}(DR_2, DR_1, \epsilon)$
`/* Compute the Rule Set similarity scores */`
foreach *rule* $r_{1i} \in P_1$ **do**
  if $r_{1i} \in PR_1$ **then**
    $rs_{1i} \leftarrow \text{ComputeRuleSimilarity}(r_{1i}, \Phi_1^P)$
  else if $r_{1i} \in DR_1$ **then**
    $rs_{1i} \leftarrow \text{ComputeRuleSimilarity}(r_{1i}, \Phi_1^D)$
end
foreach *rule* $r_{2j} \in P_2$ **do**
  if $r_{2j} \in PR_2$ **then**
    $rs_{2j} \leftarrow \text{ComputeRuleSimilarity}(r_{2j}, \Phi_2^P)$
  else if $r_{1i} \in DR_1$ **then**
    $rs_{2j} \leftarrow \text{ComputeRuleSimilarity}(r_{2j}, \Phi_2^D)$
end
$S_{rule-set}^P \leftarrow$ average of rs of permit rules
$S_{rule-set}^D \leftarrow$ average of rs of deny rules
`/* Compute the overall similarity score */`
$S_{policy}(P_1, P_2) = w_T S_T(P_1, P_2) + w_p S_{rule-set}^P + w_d S_{rule-set}^D$
return $S_{policy}(P_1, P_2)$

Algorithm A.2: $ComputePhiMapping(R', R'', \epsilon)$ Algorithm - Lin et al's Procedure for Computing Φ Mapping [12].

input : R' and R'' are sets of rules and ϵ is a threshold value.

output: Φ mapping.

foreach *rule* $r' \in R'$ **do**
    $\Phi(r') = \emptyset$
    foreach *rule* $r'' \in R''$ **do**
        if $S_{rule}(r', r'') \geq \epsilon$ **then**
            $\Phi(r') = \Phi(r') \cup \{r''\}$
        end
    end
end
return Φ

Algorithm A.3: $ComputeRuleSimilarity(r', \Phi)$ Algorithm - Lin et al's Procedure for Computing Rule Similarity [12].

input : r' is a rule and Φ is a mapping between rules.

output: Rule similarity score rs.

foreach *rule* $r'' \in \Phi$ **do**
    $sum = sum + S_{rule}(r', r'')$
end
$rs = \frac{sum}{|\Phi|}$
return rs

```
<Policy PolicyId=P1 RuleCombiningAlgId=deny-overrides>
   <PolicyTarget>
      <Subject GroupName=QPIFCollaboration>
   </PolicyTarget>
   <Obligation>
      <Obligation FulfillOn=Permit>
         <AttributeAssignment Action=mailto
                     Address=admin@qpif DataType=string>
      </Obligation>
   </Obligations>
   <RuleId=R11 Effect=Permit>
      <Target>
         <Subject Designation
                     belong_to{Manager,Researcher,Intern,TechnicalStaff}>
         <Resource FileType belong_to{Source,Documentation,Executable}>
         <Action AccessType belong_to{Read,Write}>
      </Target>
      <Condition FileSize <= 100MB>
   </Rule>
   <RuleId=R12 Effect=Deny>
      <Target>
         <Subject Designation belong_to{Intern,Researcher,TechnicalStaff}>
         <Resource FileType belong_to{Documentation}>
         <Action AccessType=Write>
      </Target>
      <Condition 19:00 <= Time <= 21:00>
   </Rule>
</Policy>
```

Figure 7: Resource Owner Policy P_1.

```
<Policy PolicyId=P2 RuleCombiningAlgId=deny-overrides>
   <PolicyTarget>
      <Subject GroupName belong_to{QPIFCollaboration, ExternalCollaboration}>
   </PolicyTarget>
   <RuleId=R21 Effect=Permit>
      <Target>
         <Subject Designation belong_to{Intern, Researcher, TechnicalStaff} >
         <Action AccessType belong to{Read, Write}>
      </Target>
      <Condition FileSize <= 120MB >
   </Rule>
   <RuleId=R22 Effect=Permit>
      <Target>
         <Subject Designation=TechnicalStaff>
         <Action AccessType belong_to{Read, Write}>
      </Target>
      <Condition 19:00 <= Time <= 22:00 >
   </Rule>
   <RuleId=R23 Effect=Deny>
      <Target>
         <Subject Designation=Intern>
         <Action AccessType=Write>
      </Target>
      <Condition {19:00 <= Time <= 22:00>
   </Rule>
   <RuleId=R24 Effect=Deny>
      <Target>
         <Subject Designation belong_to{Intern, Researcher, Staff}>
         <Resource FileType=Media>
         <Action AccessType belong to{Read, Write}>
      </Target>
   </Rule>
</Policy>
```

Figure 8: Resource Owner Policy P_2.

```
<Policy PolicyId=P3 RuleCombiningAlgId=permit-overrides>
   <PolicyTarget>
      <Subject ServerGroup=MailServer>
   </PolicyTarget>
   <RuleId=R31 Effect=Permit>
      <Target>
         <Subject Designation=AdministrationStaff>
         <Resource Host=collaboration.qpif.qut.edu.au>
         <Action AccessType belong_to{Read, Write}>
      </Target>
      <Condition 8:00 <= Time <= 17:00>
   </Rule>
</Policy>
```

Figure 9: Resource Owner Policy P_3.

```
<Policy PolicyId=P4 RuleCombiningAlgId=permit-overrides>
   <PolicyTarget>
      <Subject GroupName=QPIFCollaboration>
   </PolicyTarget>
   <RuleId=R41 Effect=Permit>
      <Target>
         <Subject Designation belong_to{Researcher}>
         <Resource Project=ProjectABC>
         <Action AccessType belong_to{Read, Write}>
      </Target>
      <Condition 8:00 <= Time <= 17:00>
   </Rule>
</Policy>
```

Figure 10: Resource Owner Policy P_4.

```
<Policy PolicyId=P5 RuleCombiningAlgId=deny-overrides>
   <PolicyTarget>
      <Subject GroupName=QPIFCollaboration>
   </PolicyTarget>
   <RuleId=R51 Effect=Deny>
      <Target>
         <Subject Designation belong_to {Researcher}>
         <Resource Project=ProjectABC>
         <Action AccessType belong_to{Read, Write}>
      </Target>
      <Condition 8:00 <= Time <= 17:00>
   </Rule>
</Policy>
```

Figure 11: Resource Owner Policy P_5.